JAWAS AND SPRING

Latest Java + Spring Framework

Marcin Baranowski
Snowflake

Developers are masochist

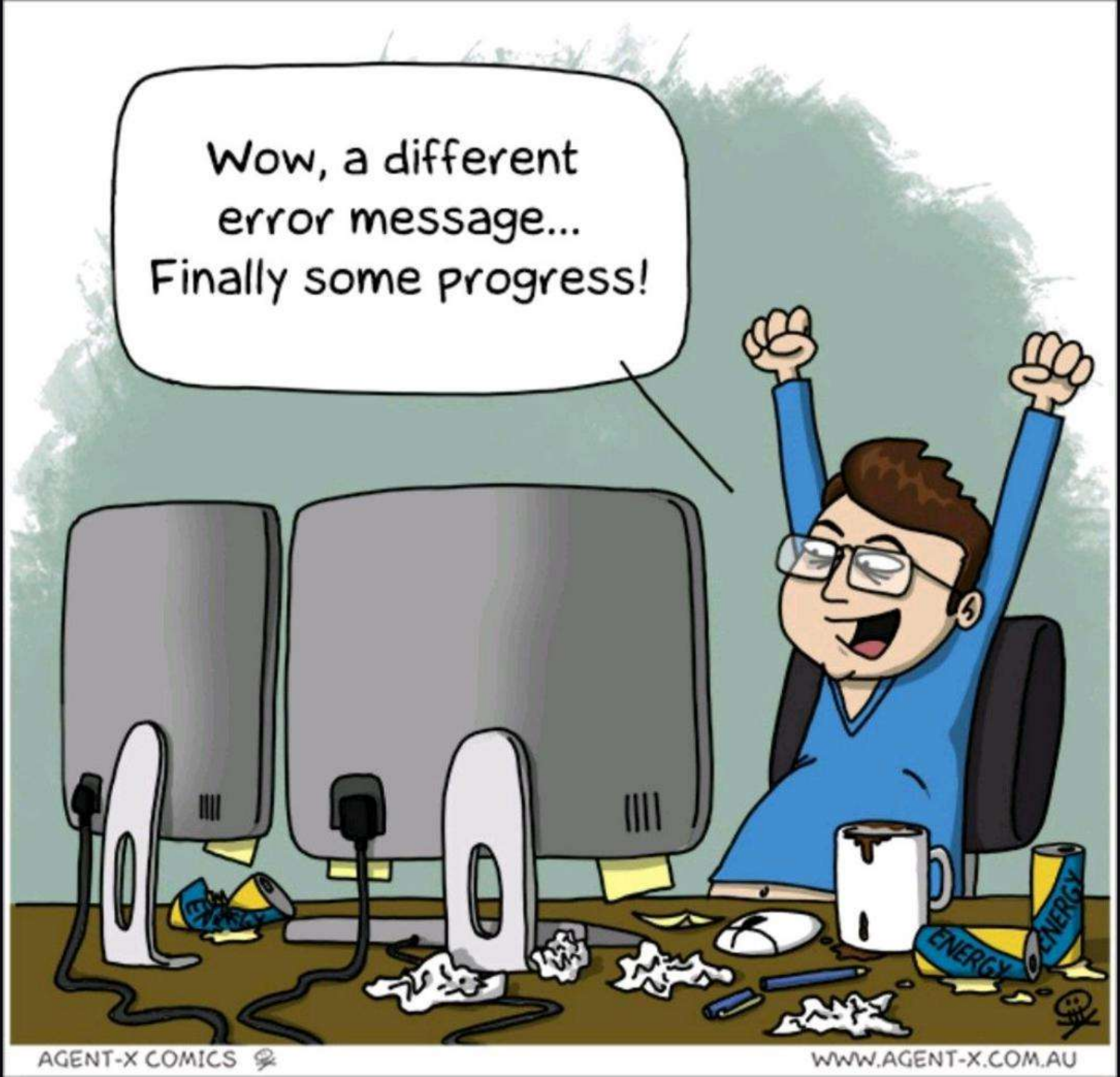
Compilation errors

Nonpassing tests

Runtime errors

Customer

Icons by https://pl.freepik.com/rawpixel-com


Wow, a different error message... Finally some progress!
ENERGY
ENERGY
ENERGY
AGENT-X COMICS
WWW.AGENT-X.COM.AU

20.6 sec

Source: https://api.stackexchange.com/docs/info#filter=default&site=stackoverflow&run=true

What if code could speak?

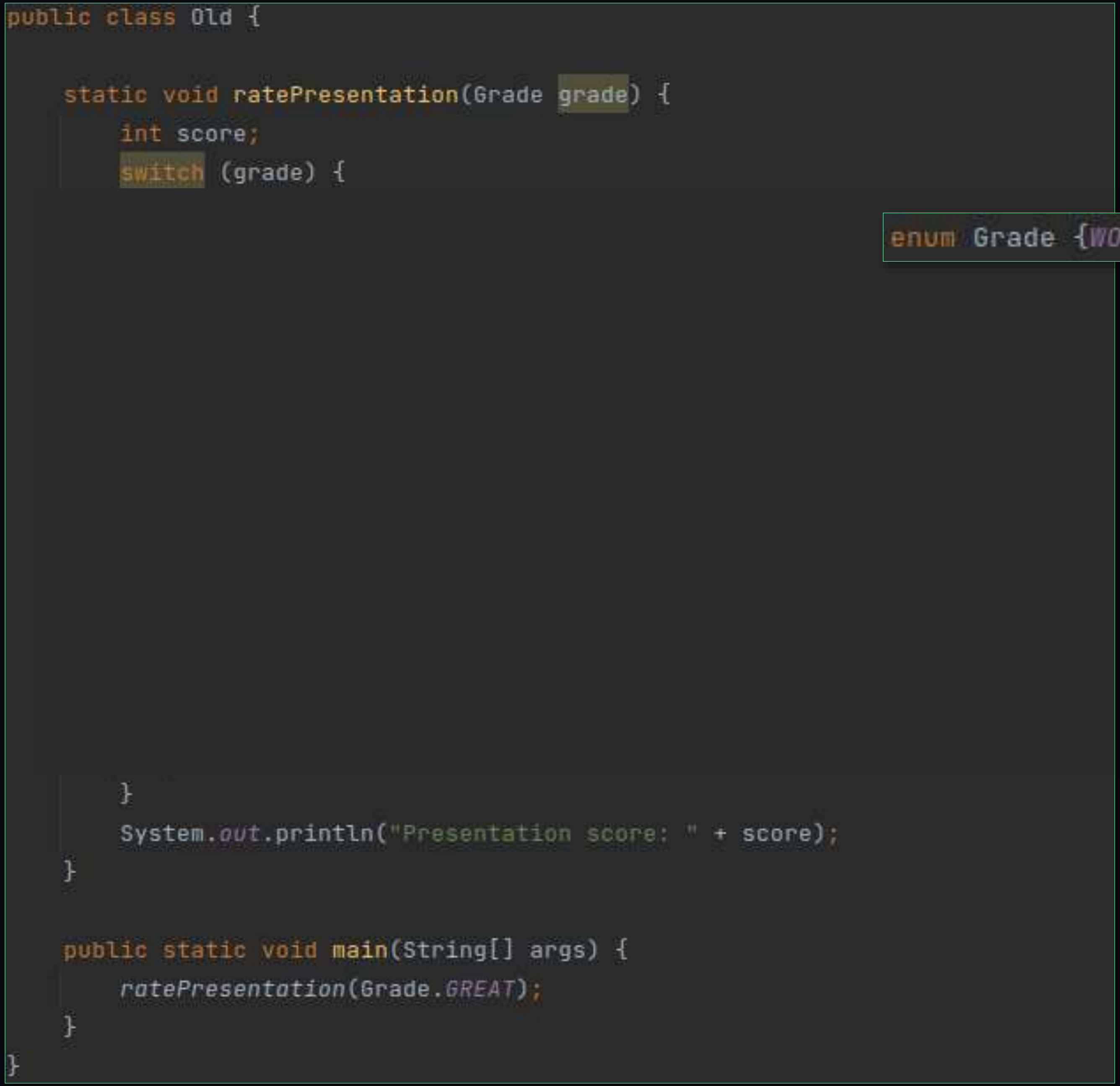
What would it tell about You?

Switch

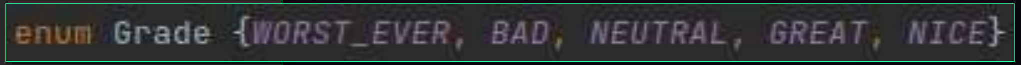
```
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```

```java
public class Old {

    static void ratePresentation(Grade grade) {
        int score;
















        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

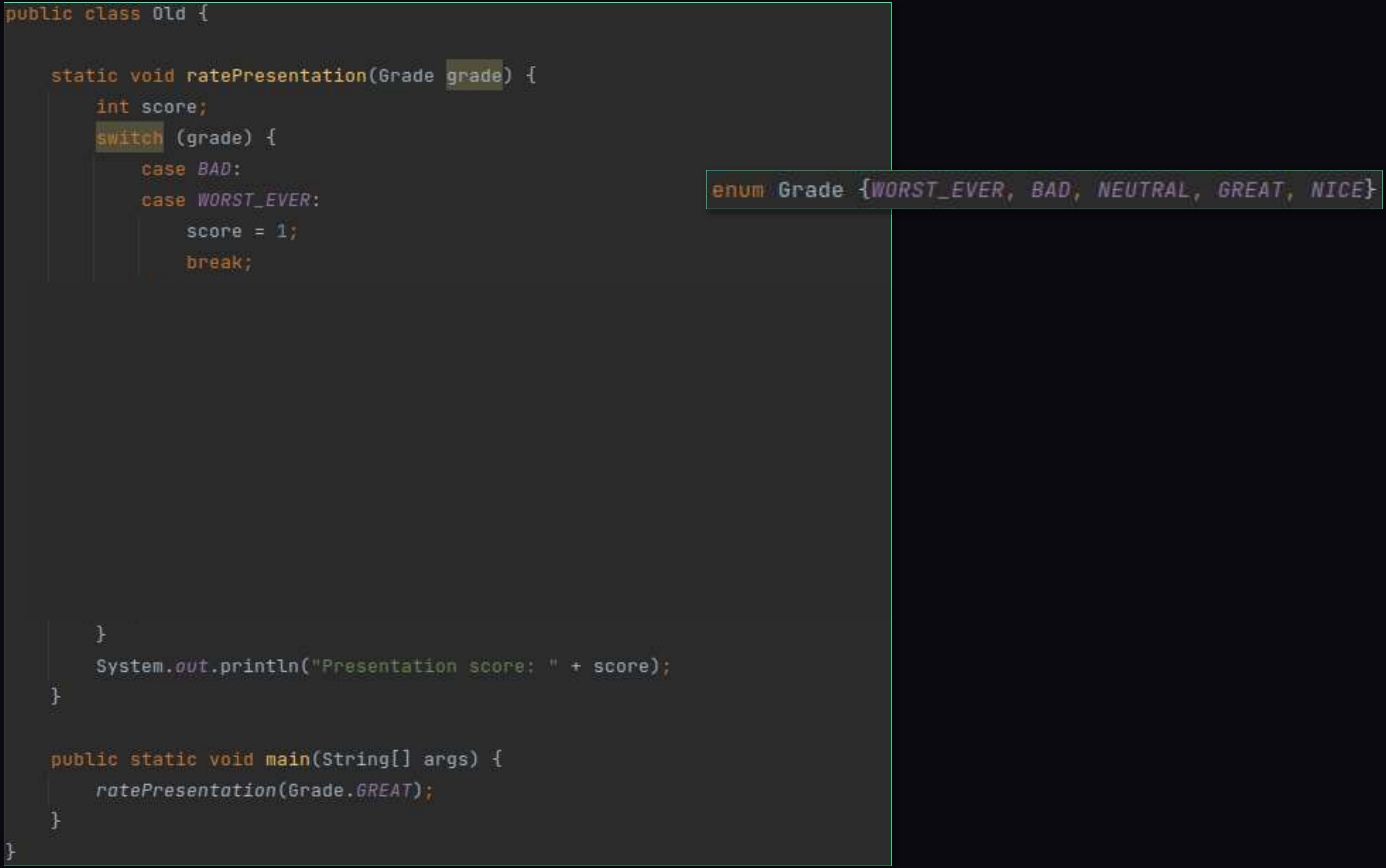
```java
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```



```java
public class Old {

    static void ratePresentation(Grade grade) {
        int score;
        switch (grade) {















        }
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

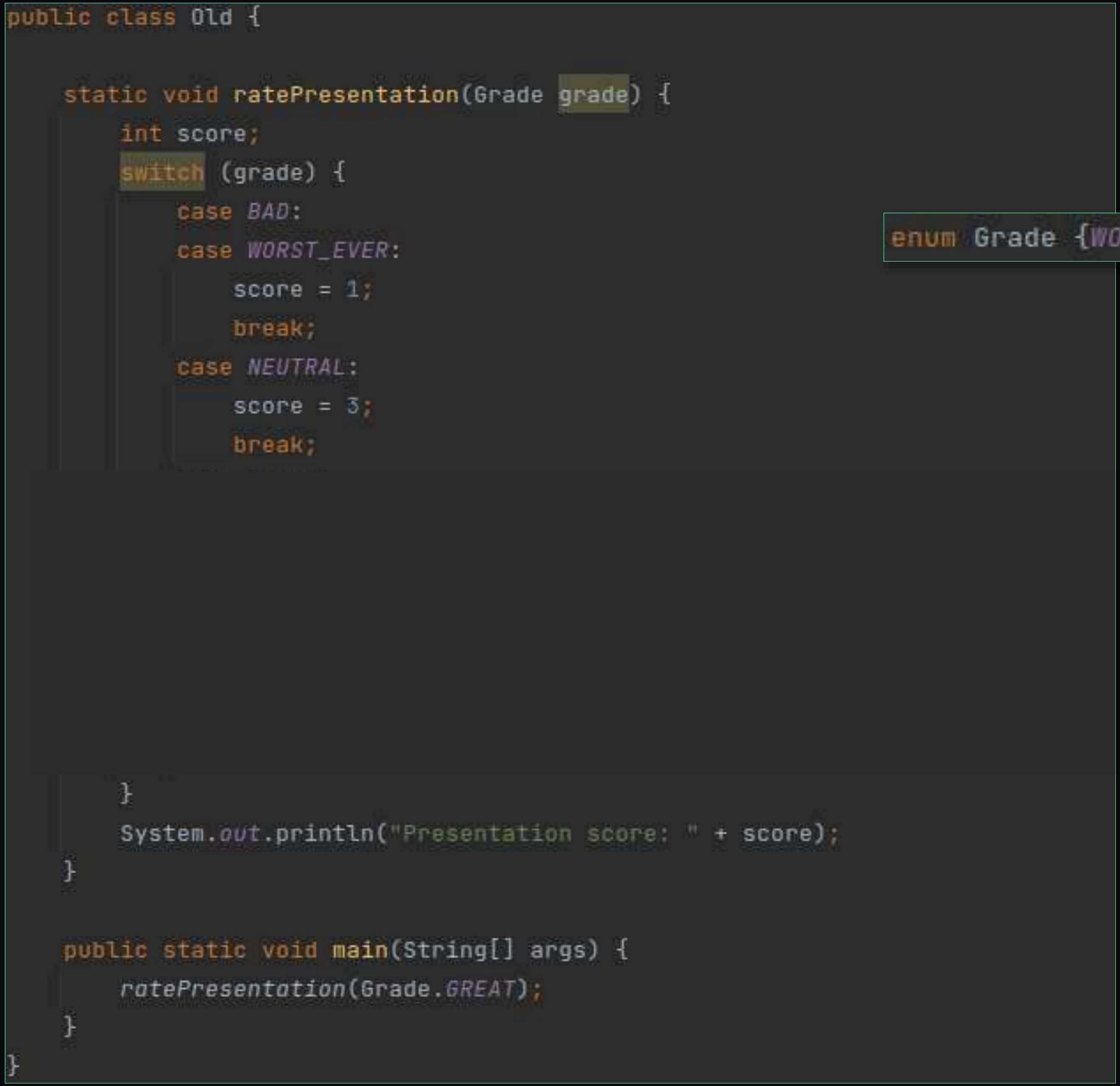


```java
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```



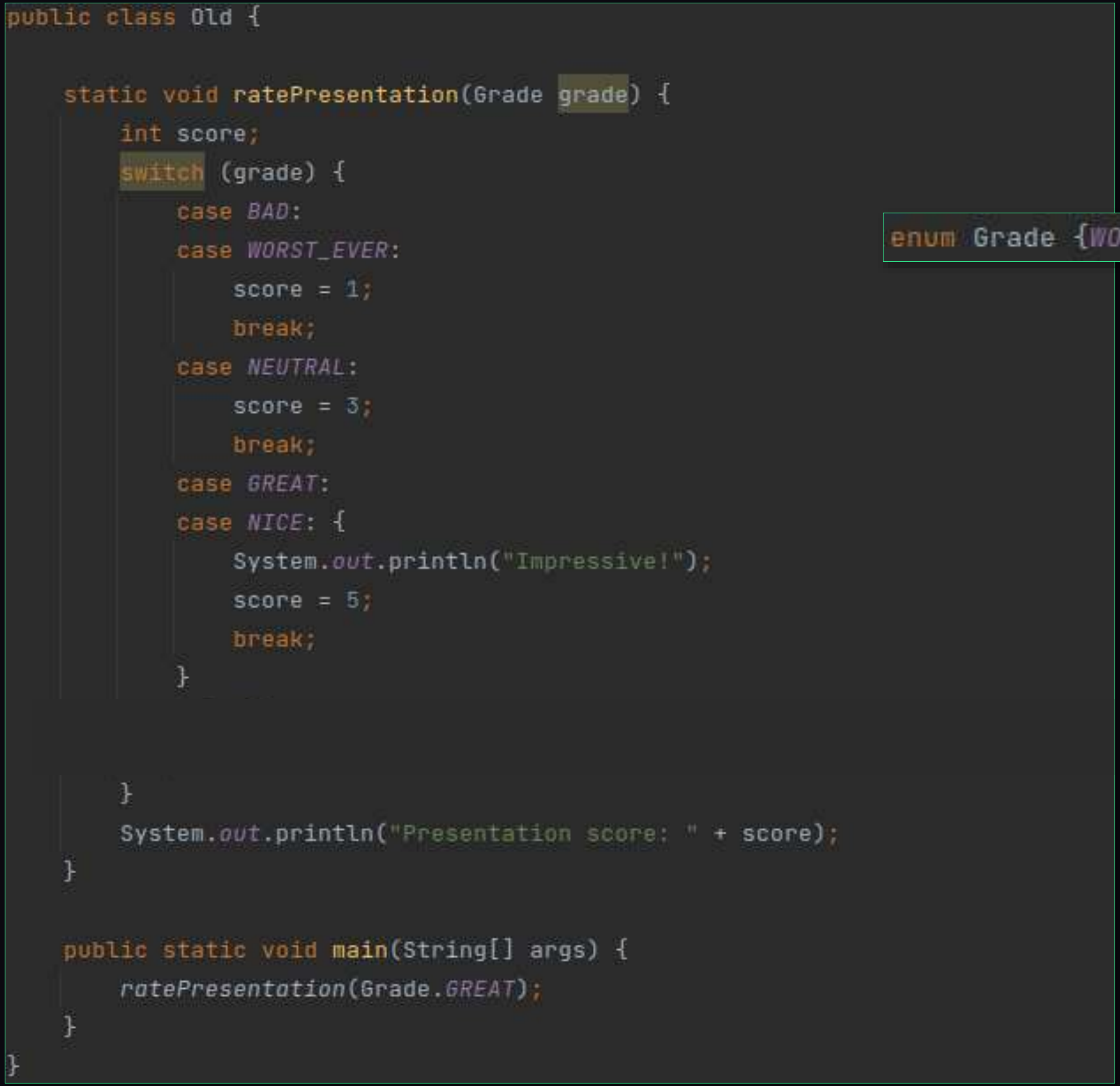
```java
public class Old {

    static void ratePresentation(Grade grade) {
        int score;
        switch (grade) {
            case BAD:
            case WORST_EVER:
                score = 1;
                break;








        }
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

```java
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```

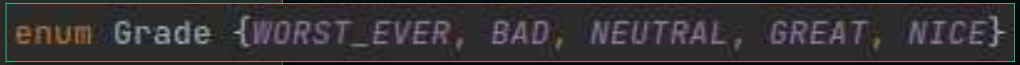


```java
public class Old {

    static void ratePresentation(Grade grade) {
        int score;
        switch (grade) {
            case BAD:
            case WORST_EVER:
                score = 1;
                break;
            case NEUTRAL:
                score = 3;
                break;









        }
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

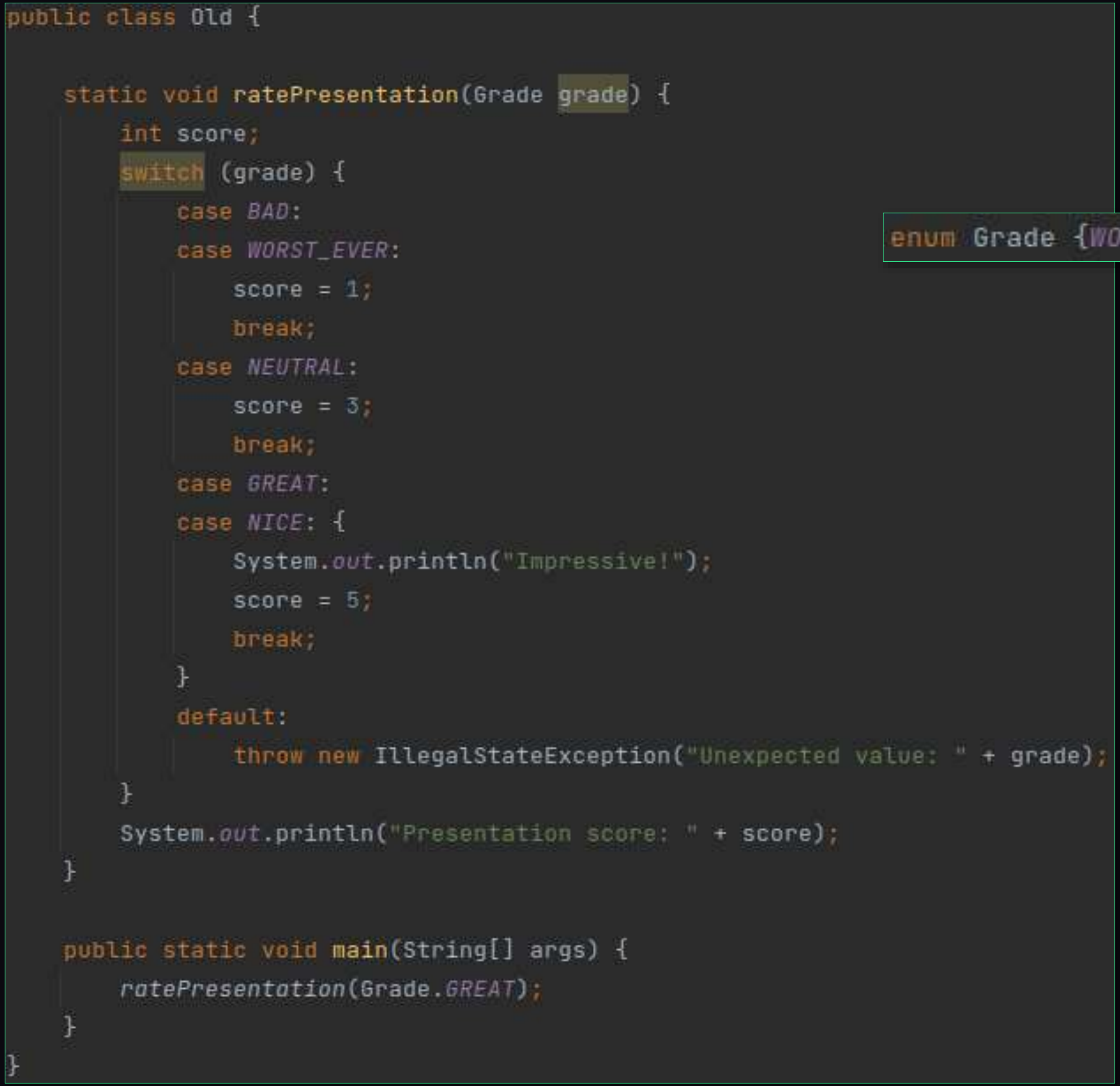
```java
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```



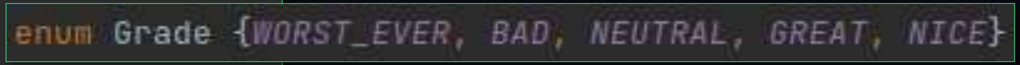
```java
public class Old {

    static void ratePresentation(Grade grade) {
        int score;
        switch (grade) {
            case BAD:
            case WORST_EVER:
                score = 1;
                break;
            case NEUTRAL:
                score = 3;
                break;
            case GREAT:
            case NICE: {
                System.out.println("Impressive!");
                score = 5;
                break;
            }


        }
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```



```java
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```



```java
public class Old {

    static void ratePresentation(Grade grade) {
        int score;
        switch (grade) {
            case BAD:
            case WORST_EVER:
                score = 1;
                break;
            case NEUTRAL:
                score = 3;
                break;
            case GREAT:
            case NICE: {
                System.out.println("Impressive!");
                score = 5;
                break;
            }
            default:
                throw new IllegalStateException("Unexpected value: " + grade);
        }
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

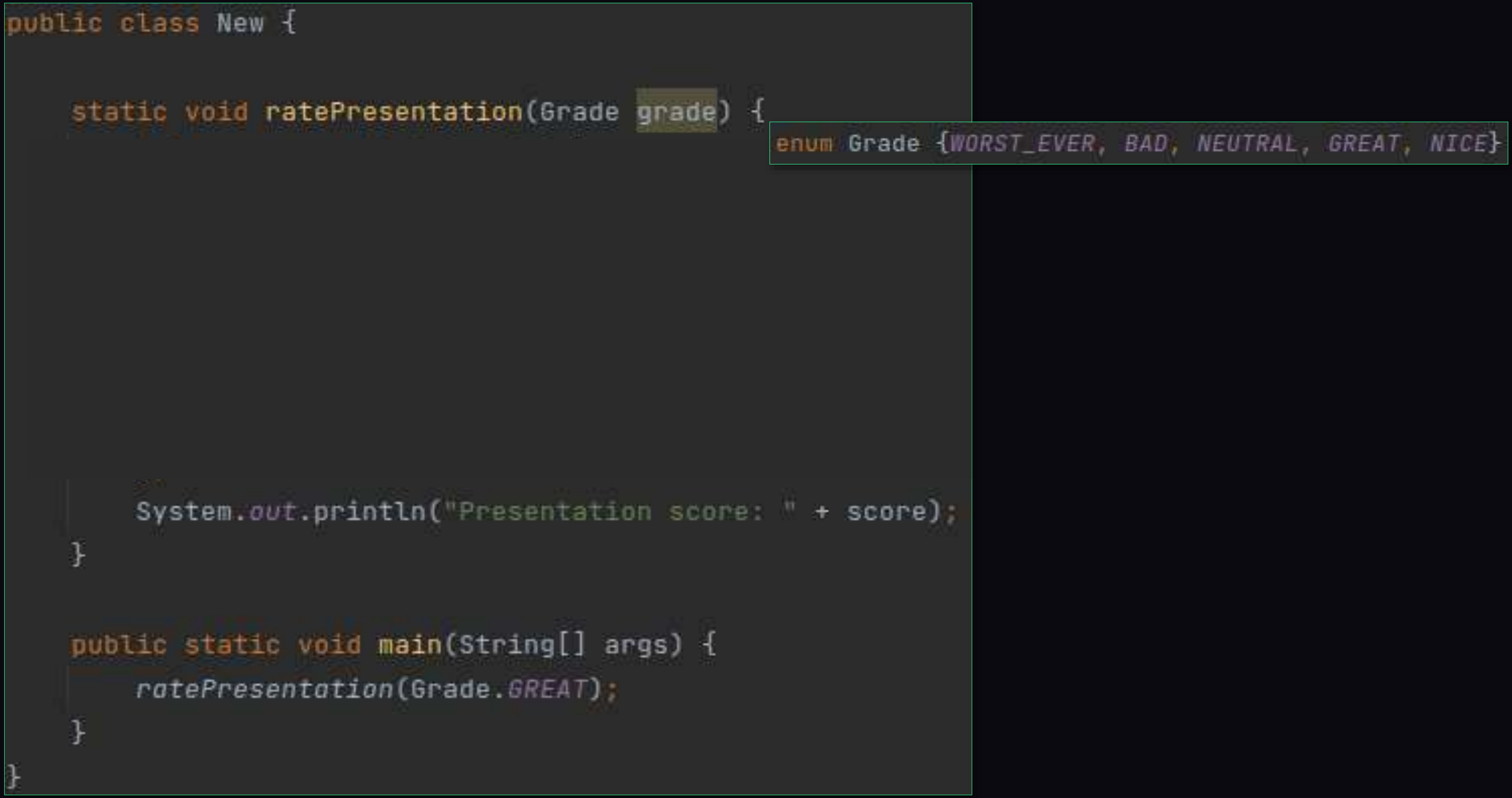


```java
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```

SWITCH EXPRESSIONS

New way

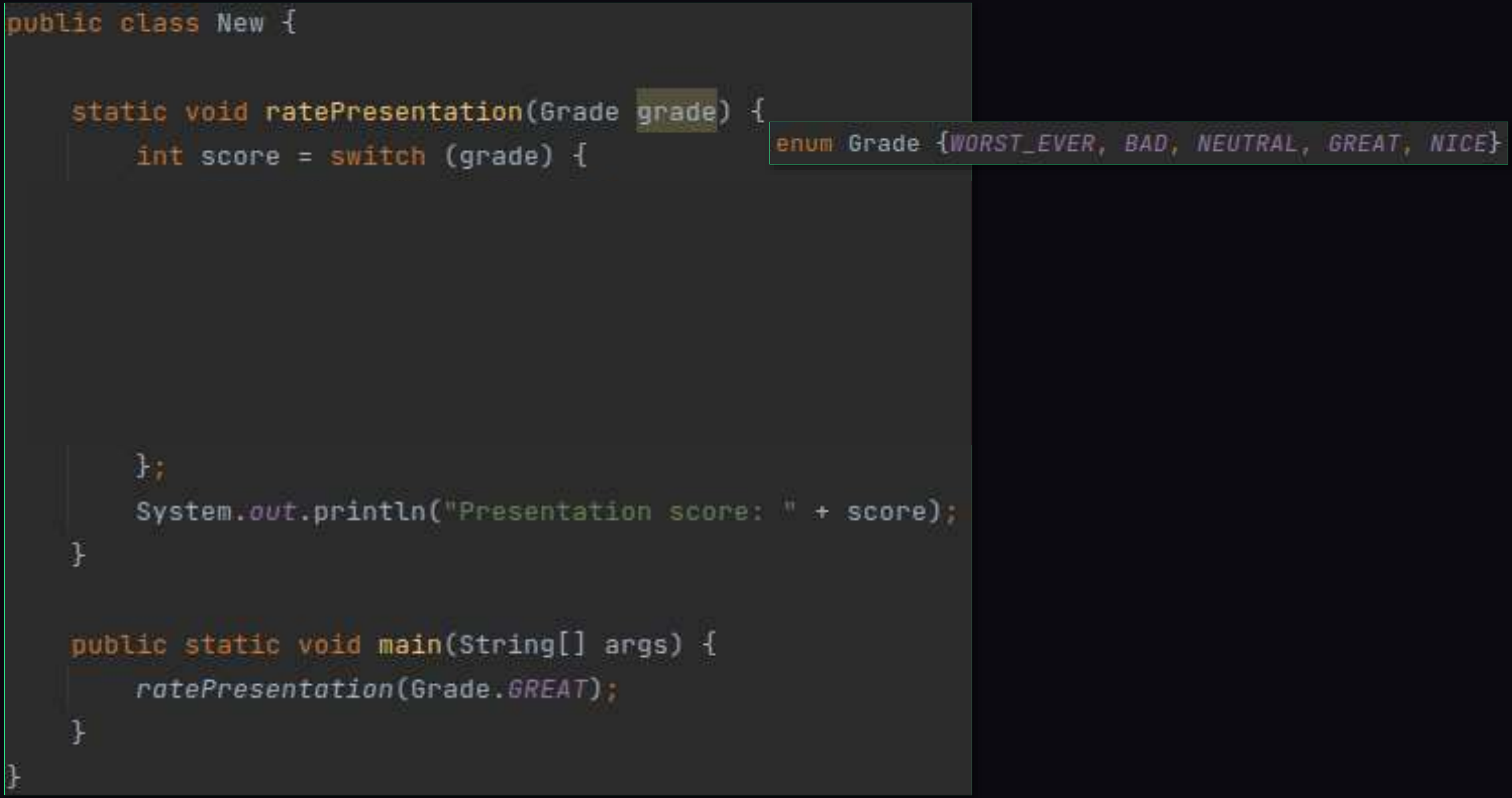
Image: deviantart.com



```java
public class New {

    static void ratePresentation(Grade grade) {









        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

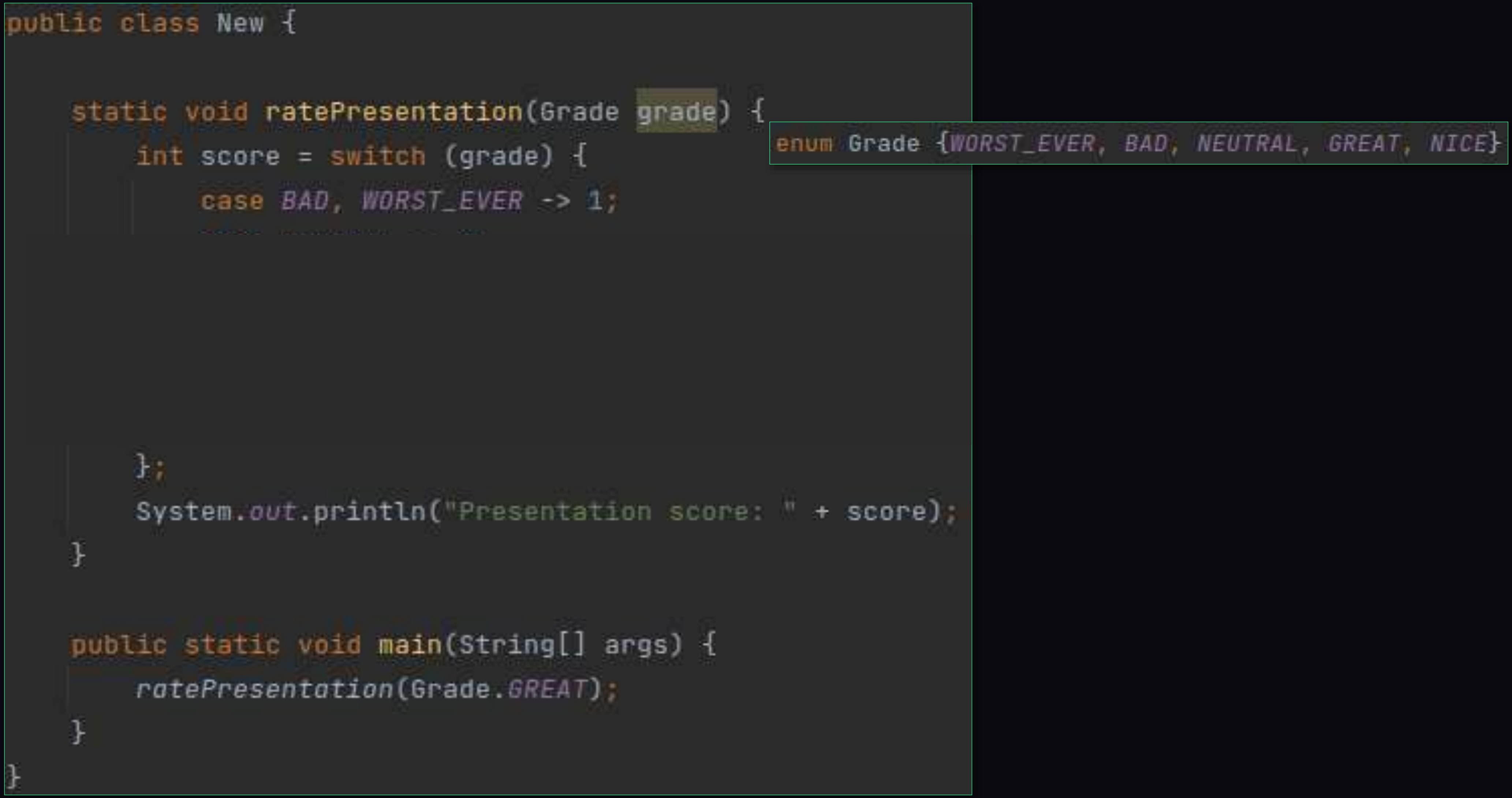
```java
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```

JDK 14



```
public class New {

    static void ratePresentation(Grade grade) {
        int score = switch (grade) {







        };
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

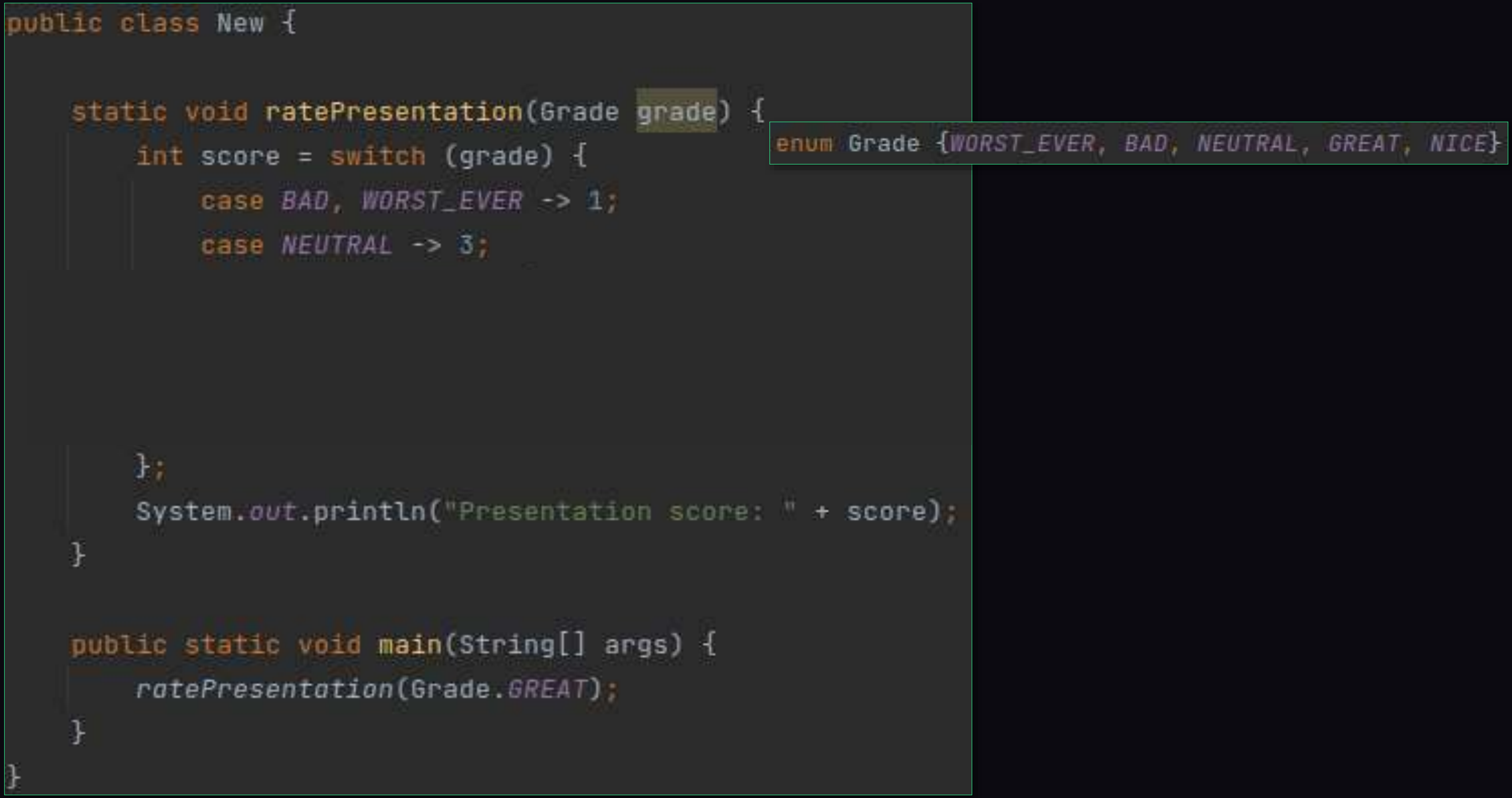
```
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```



```
public class New {

    static void ratePresentation(Grade grade) {
        int score = switch (grade) {
            case BAD, WORST_EVER -> 1;







        };
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

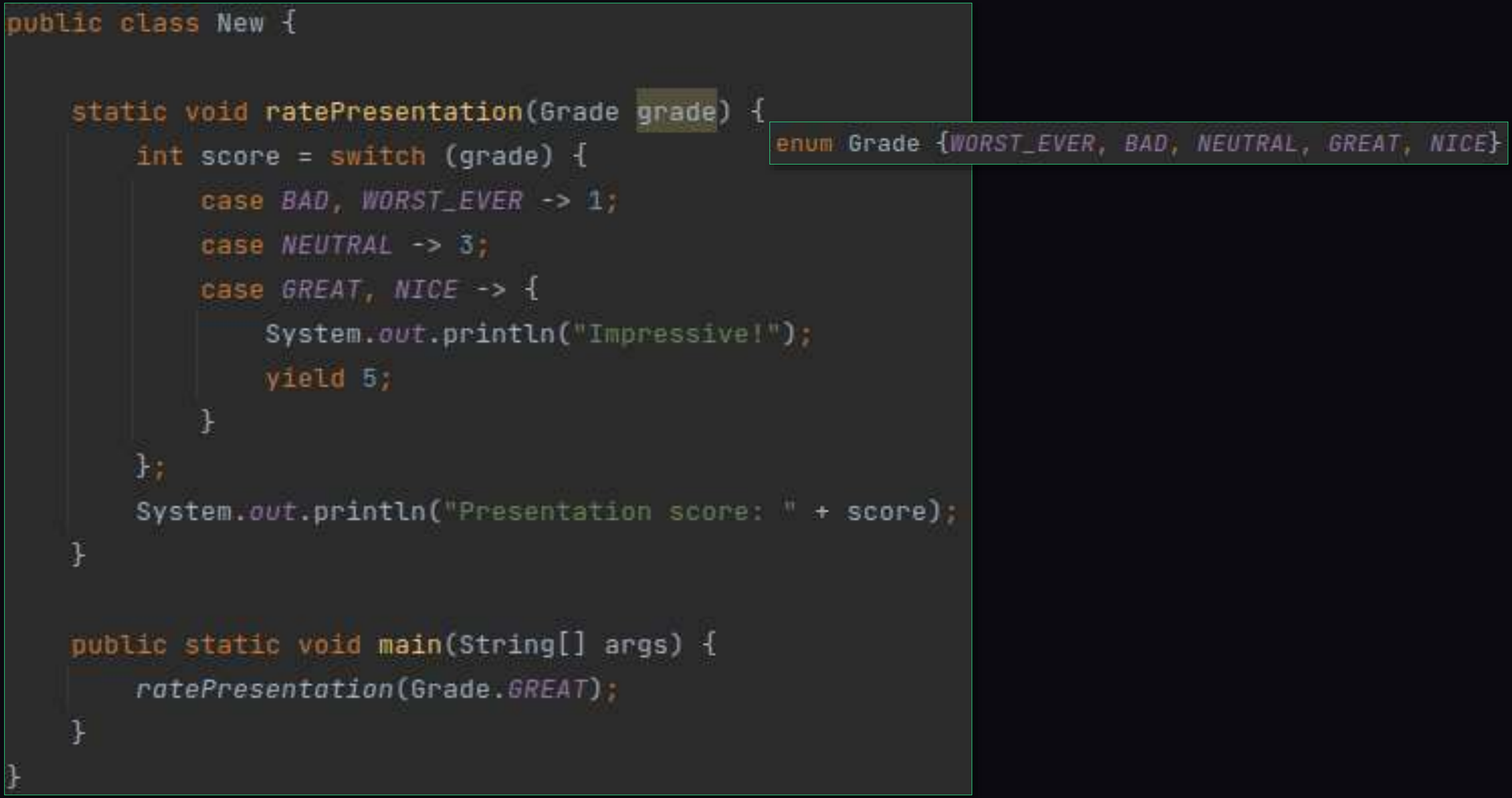
```
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```

JDK 14



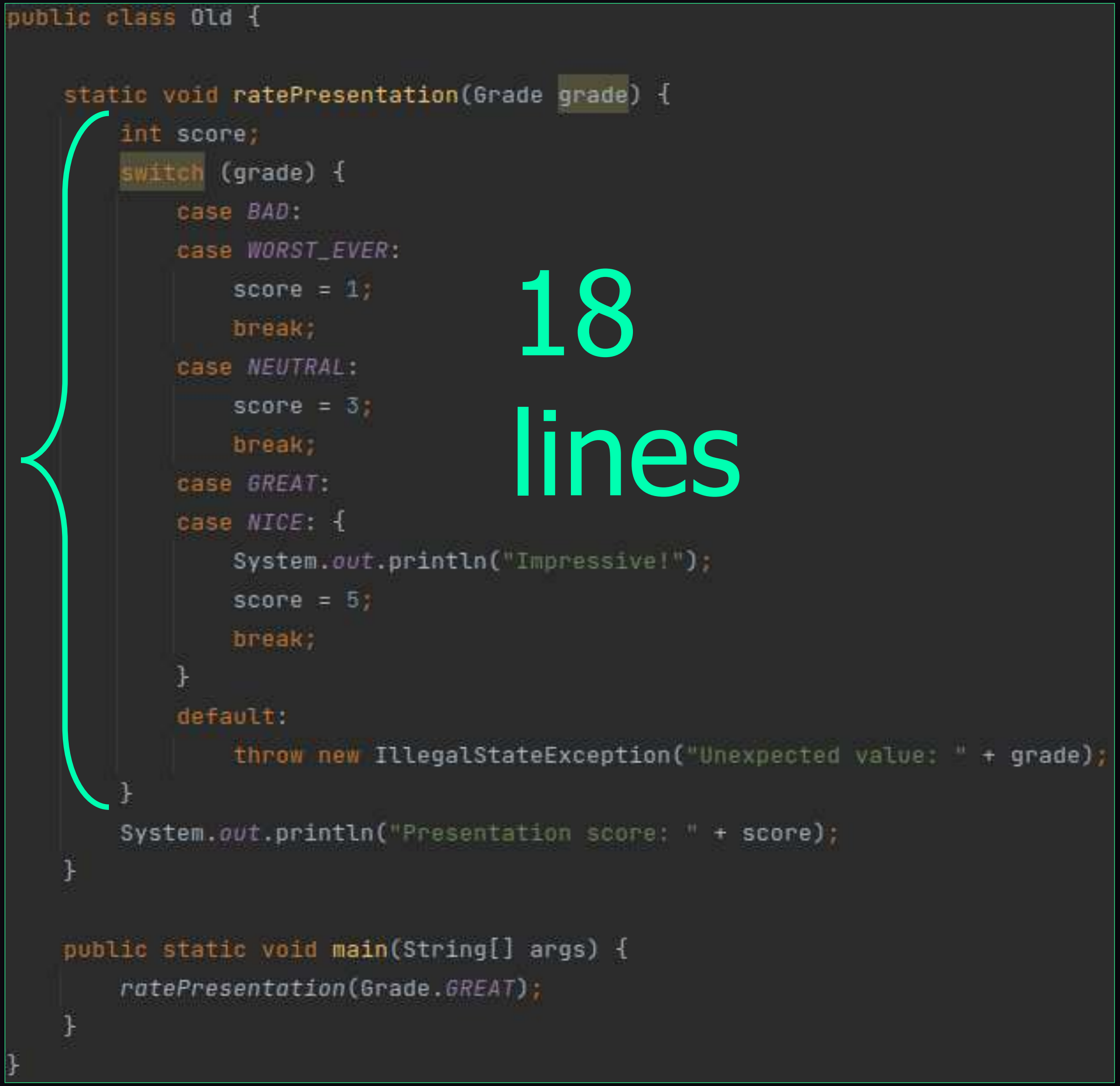
```
public class New {

    static void ratePresentation(Grade grade) {
        int score = switch (grade) {
            case BAD, WORST_EVER -> 1;
            case NEUTRAL -> 3;




        };
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

```
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```

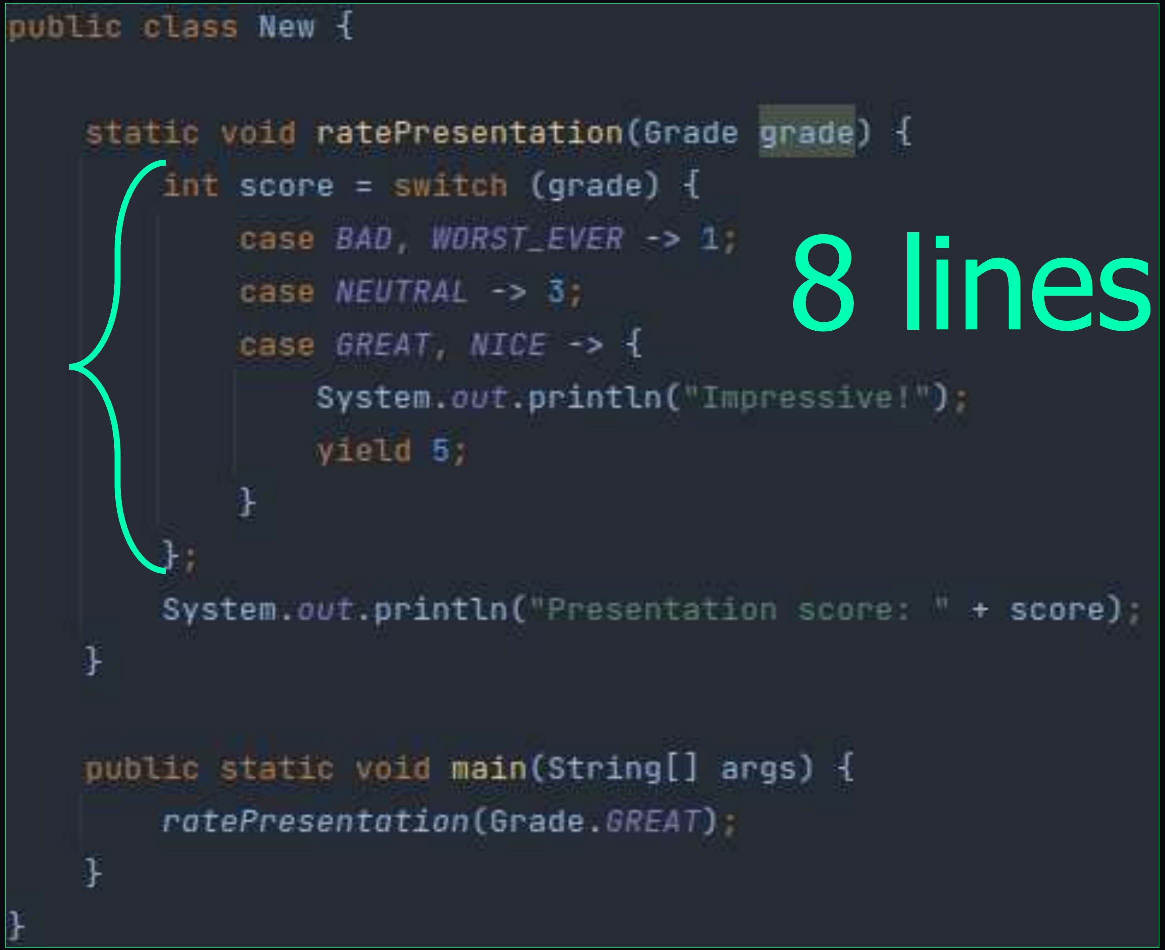


```
public class New {

    static void ratePresentation(Grade grade) {
        int score = switch (grade) {
            case BAD, WORST_EVER -> 1;
            case NEUTRAL -> 3;
            case GREAT, NICE -> {
                System.out.println("Impressive!");
                yield 5;
            }
        };
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

```
enum Grade {WORST_EVER, BAD, NEUTRAL, GREAT, NICE}
```



```java
public class Old {

    static void ratePresentation(Grade grade) {
        int score;
        switch (grade) {
            case BAD:
            case WORST_EVER:
                score = 1;
                break;
            case NEUTRAL:
                score = 3;
                break;
            case GREAT:
            case NICE: {
                System.out.println("Impressive!");
                score = 5;
                break;
            }
            default:
                throw new IllegalStateException("Unexpected value: " + grade);
        }
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```



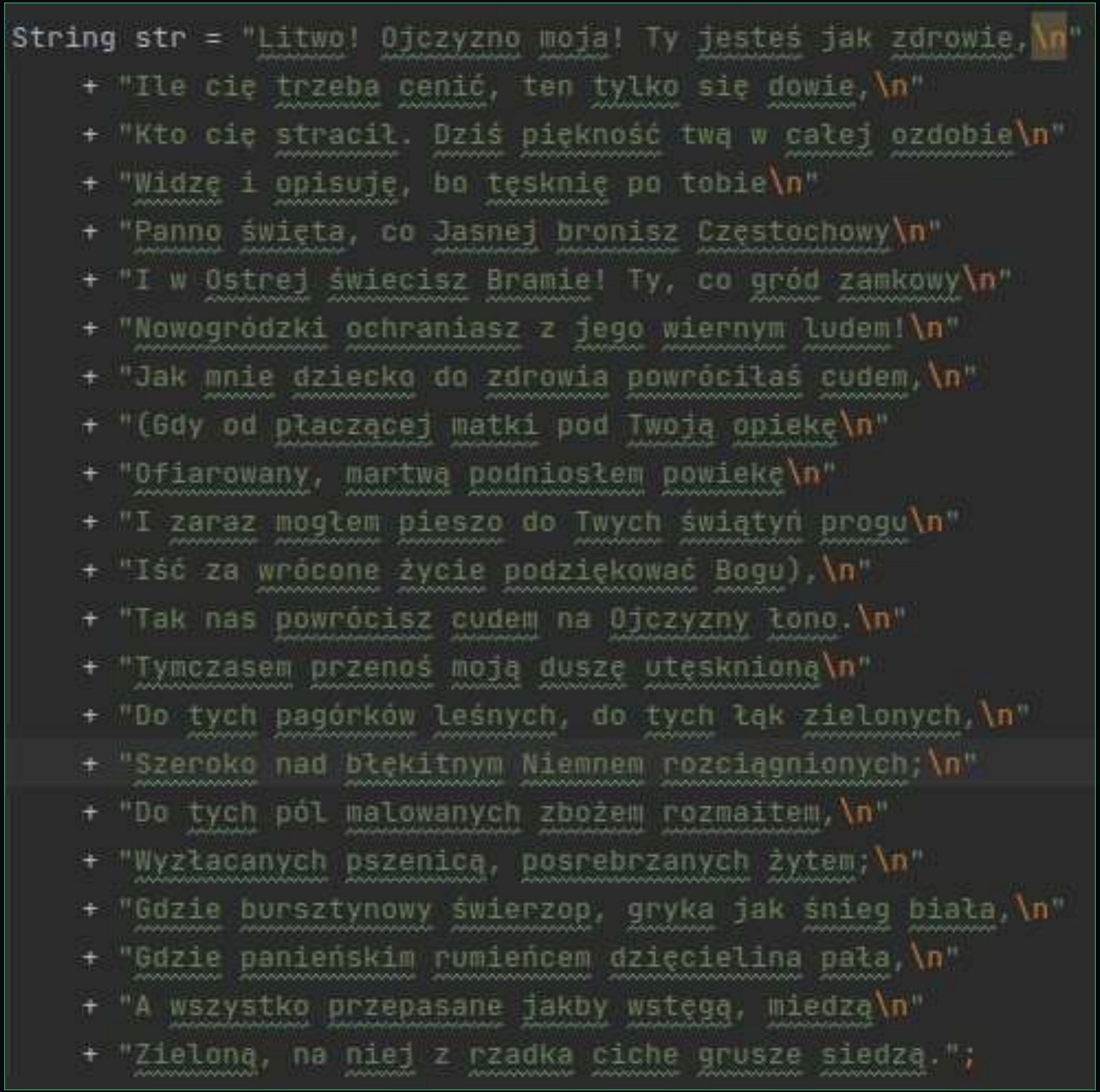
```java
public class New {

    static void ratePresentation(Grade grade) {
        int score = switch (grade) {
            case BAD, WORST_EVER -> 1;
            case NEUTRAL -> 3;
            case GREAT, NICE -> {
                System.out.println("Impressive!");
                yield 5;
            }
        };
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

How many planets are there in the solar system?

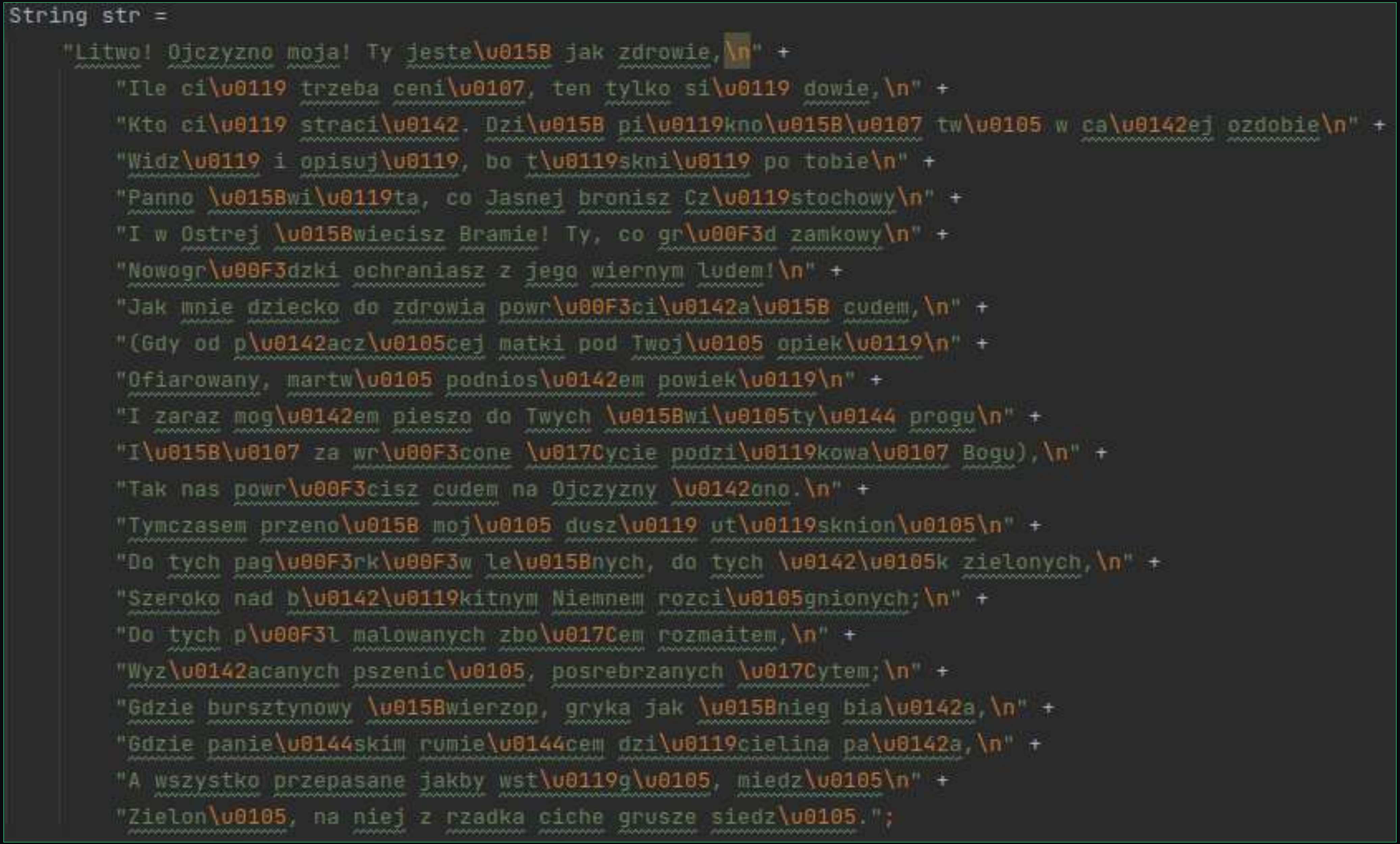
REMOVAL OF NASHORN JAVASCRIPT

Image: deviantart.com

Do You need PhD to write multiline strings in Java?



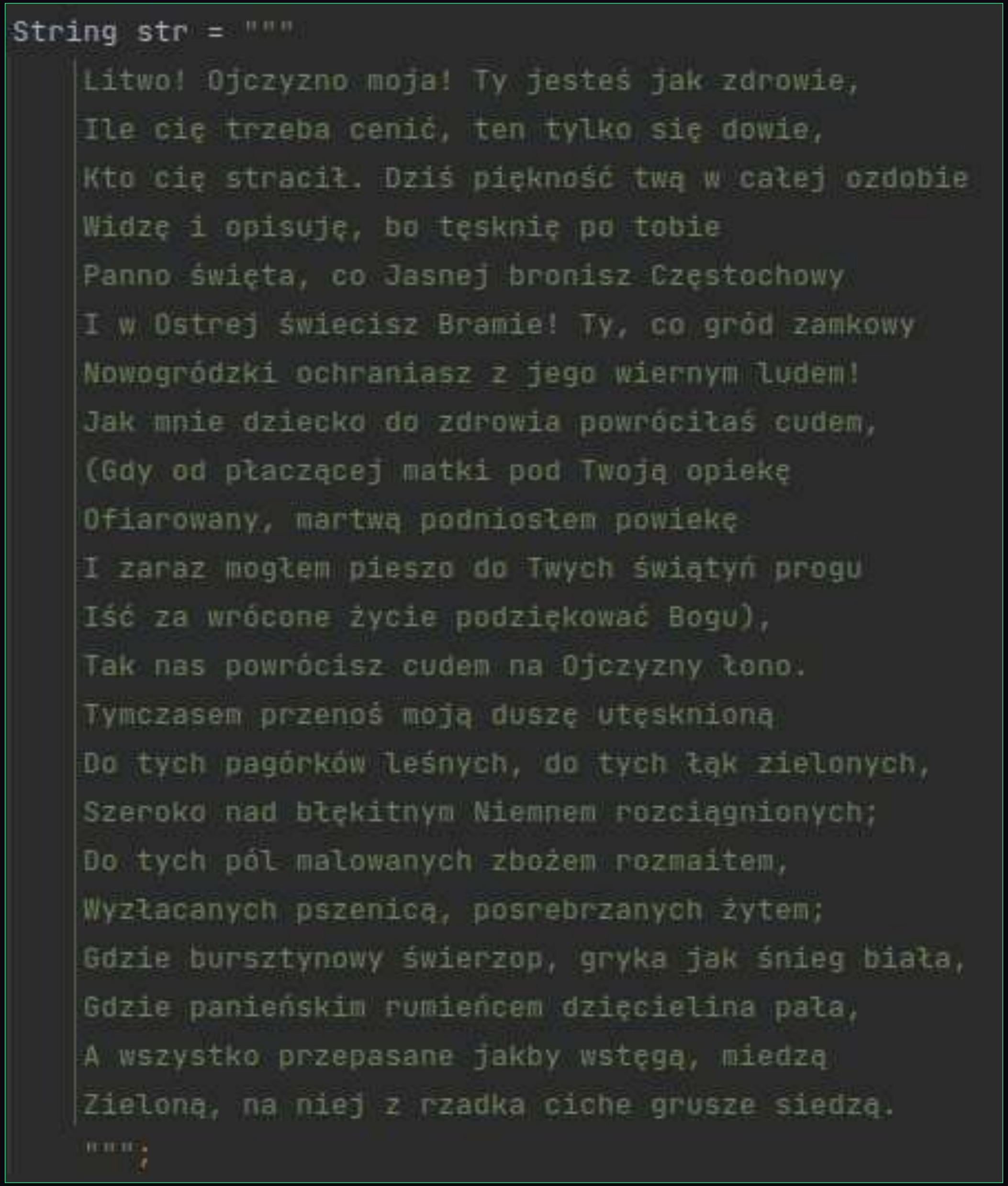
```
String str = "Litwo! Ojczyzno moja! Ty jesteś jak zdrowie,\n"
        + "Ile cię trzeba cenić, ten tylko się dowie,\n"
        + "Kto cię stracił. Dziś piękność twą w całej ozdobie\n"
        + "Widzę i opisuję, bo tęsknię po tobie\n"
        + "Panno święta, co Jasnej bronisz Częstochowy\n"
        + "I w Ostrej świecisz Bramie! Ty, co gród zamkowy\n"
        + "Nowogródzki ochraniasz z jego wiernym ludem!\n"
        + "Jak mnie dziecko do zdrowia powróciłaś cudem,\n"
        + "(Gdy od płaczącej matki pod Twoją opiekę\n"
        + "Ofiarowany, martwą podniosłem powiekę\n"
        + "I zaraz mogłem pieszo do Twych świątyń progu\n"
        + "Iść za wrócone życie podziękować Bogu),\n"
        + "Tak nas powrócisz cudem na Ojczyzny łono.\n"
        + "Tymczasem przenoś moją duszę utęsknioną\n"
        + "Do tych pagórków leśnych, do tych łąk zielonych,\n"
        + "Szeroko nad błękitnym Niemnem rozciągnionych;\n"
        + "Do tych pól malowanych zbożem rozmaitem,\n"
        + "Wyzłacanych pszenicą, posrebrzanych żytem;\n"
        + "Gdzie bursztynowy świerzop, gryka jak śnieg biała,\n"
        + "Gdzie panieńskim rumieńcem dzięcielina pała,\n"
        + "A wszystko przepasane jakby wstęgą, miedzą\n"
        + "Zieloną, na niej z rzadka ciche grusze siedzą.";
```



```java
String str =
    "Litwo! Ojczyzno moja! Ty jeste\u015B jak zdrowie,\n" +
        "Ile ci\u0119 trzeba ceni\u0107, ten tylko si\u0119 dowie,\n" +
        "Kto ci\u0119 straci\u0142. Dzi\u015B pi\u0119kno\u015B\u0107 tw\u0105 w ca\u0142ej ozdobie\n" +
        "Widz\u0119 i opisuj\u0119, bo t\u0119skni\u0119 po tobie\n" +
        "Panno \u015Bwi\u0119ta, co Jasnej bronisz Cz\u0119stochowy\n" +
        "I w Ostrej \u015Bwiecisz Bramie! Ty, co gr\u00F3d zamkowy\n" +
        "Nowogr\u00F3dzki ochraniasz z jego wiernym ludem!\n" +
        "Jak mnie dziecko do zdrowia powr\u00F3ci\u0142a\u015B cudem,\n" +
        "(Gdy od p\u0142acz\u0105cej matki pod Twoj\u0105 opiek\u0119\n" +
        "Ofiarowany, martw\u0105 podnios\u0142em powiek\u0119\n" +
        "I zaraz mog\u0142em pieszo do Twych \u015Bwi\u0105ty\u0144 progu\n" +
        "I\u015B\u0107 za wr\u00F3cone \u017Cycie podzi\u0119kowa\u0107 Bogu),\n" +
        "Tak nas powr\u00F3cisz cudem na Ojczyzny \u0142ono.\n" +
        "Tymczasem przeno\u015B moj\u0105 dusz\u0119 ut\u0119sknion\u0105\n" +
        "Do tych pag\u00F3rk\u00F3w le\u015Bnych, do tych \u0142\u0105k zielonych,\n" +
        "Szeroko nad b\u0142\u0119kitnym Niemnem rozci\u0105gnionych;\n" +
        "Do tych p\u00F3l malowanych zbo\u017Cem rozmaitem,\n" +
        "Wyz\u0142acanych pszenic\u0105, posrebrzanych \u017Cytem;\n" +
        "Gdzie bursztynowy \u015Bwierzop, gryka jak \u015Bnieg bia\u0142a,\n" +
        "Gdzie panie\u0144skim rumie\u0144cem dzi\u0119cielina pa\u0142a,\n" +
        "A wszystko przepasane jakby wst\u0119g\u0105, miedz\u0105\n" +
        "Zielon\u0105, na niej z rzadka ciche grusze siedz\u0105.";
```

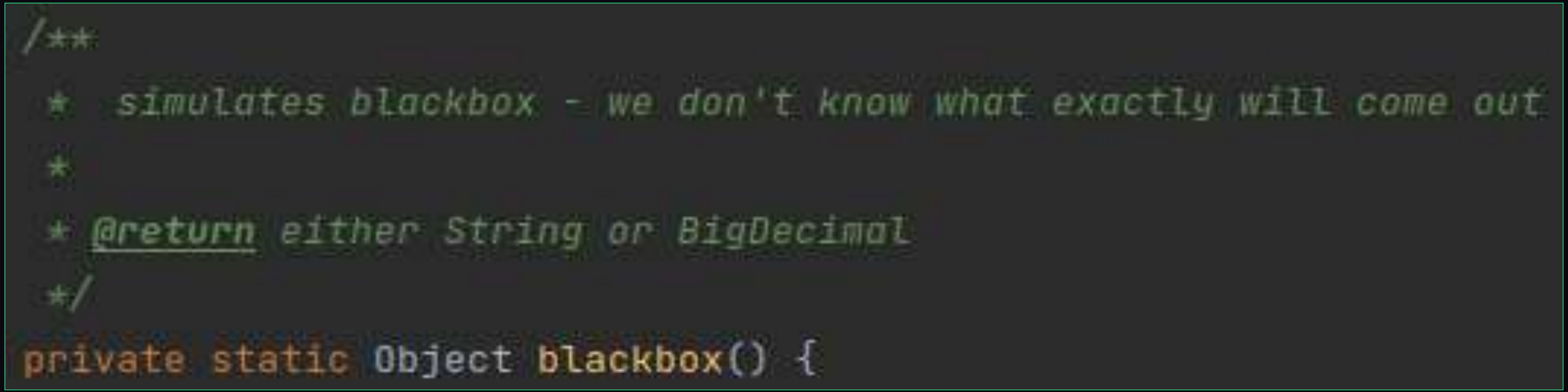
TEXTBLOCKS

JDK 15

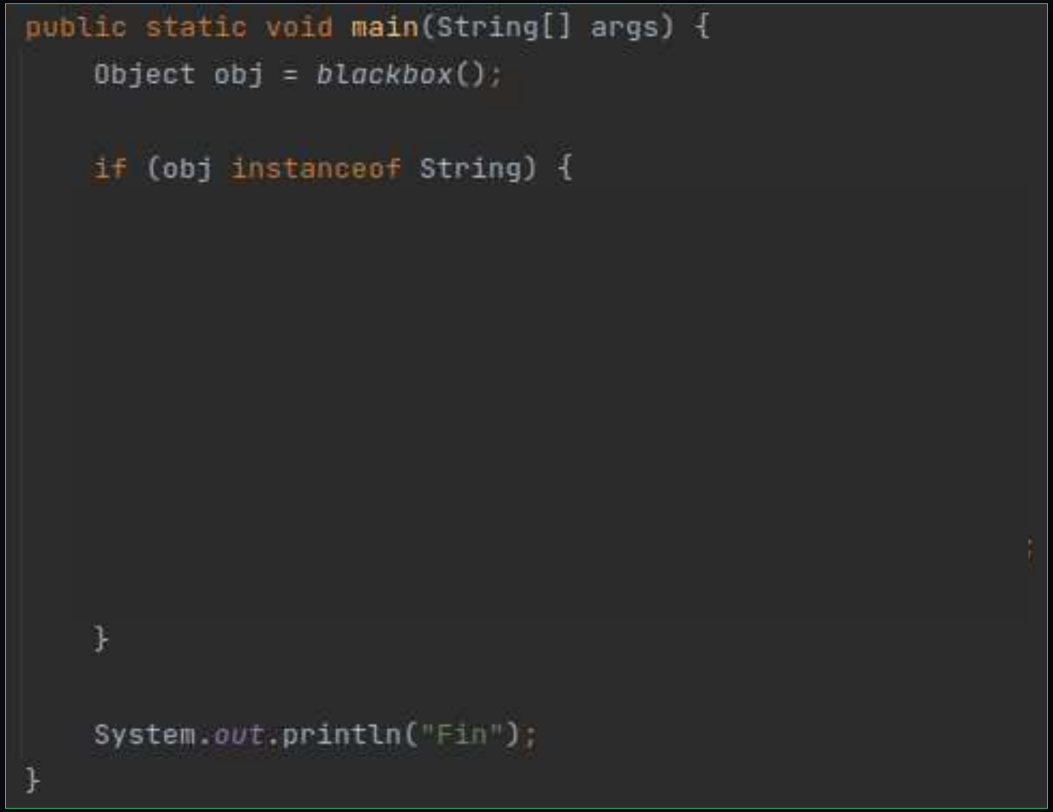


```java
String str = """
    Litwo! Ojczyzno moja! Ty jesteś jak zdrowie,
    Ile cię trzeba cenić, ten tylko się dowie,
    Kto cię stracił. Dziś piękność twą w całej ozdobie
    Widzę i opisuję, bo tęsknię po tobie
    Panno święta, co Jasnej bronisz Częstochowy
    I w Ostrej świecisz Bramie! Ty, co gród zamkowy
    Nowogródzki ochraniasz z jego wiernym ludem!
    Jak mnie dziecko do zdrowia powróciłaś cudem,
    (Gdy od płaczącej matki pod Twoją opiekę
    Ofiarowany, martwą podniosłem powiekę
    I zaraz mogłem pieszo do Twych świątyń progu
    Iść za wrócone życie podziękować Bogu),
    Tak nas powrócisz cudem na Ojczyzny łono.
    Tymczasem przenoś moją duszę utęsknioną
    Do tych pagórków leśnych, do tych łąk zielonych,
    Szeroko nad błękitnym Niemnem rozciągnionych;
    Do tych pól malowanych zbożem rozmaitem,
    Wyzłacanych pszenicą, posrebrzanych żytem;
    Gdzie bursztynowy świerzop, gryka jak śnieg biała,
    Gdzie panieńskim rumieńcem dzięcielina pała,
    A wszystko przepasane jakby wstęgą, miedzą
    Zieloną, na niej z rzadka ciche grusze siedzą.
    """;
```

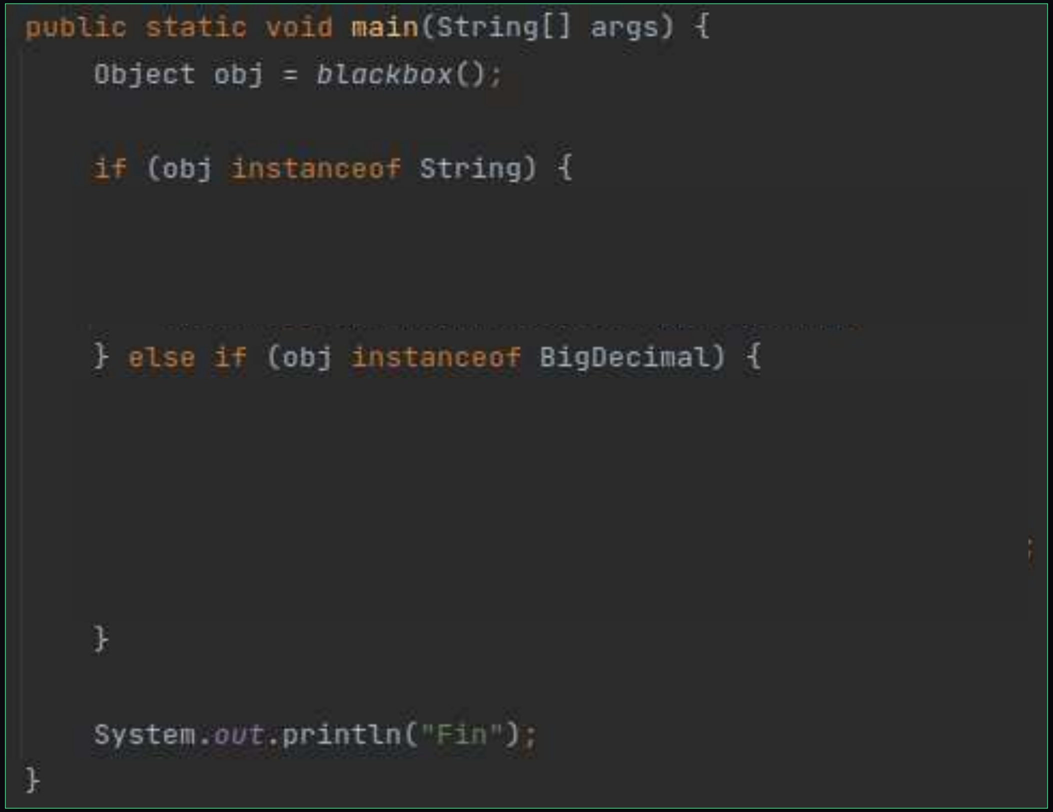
instanceof



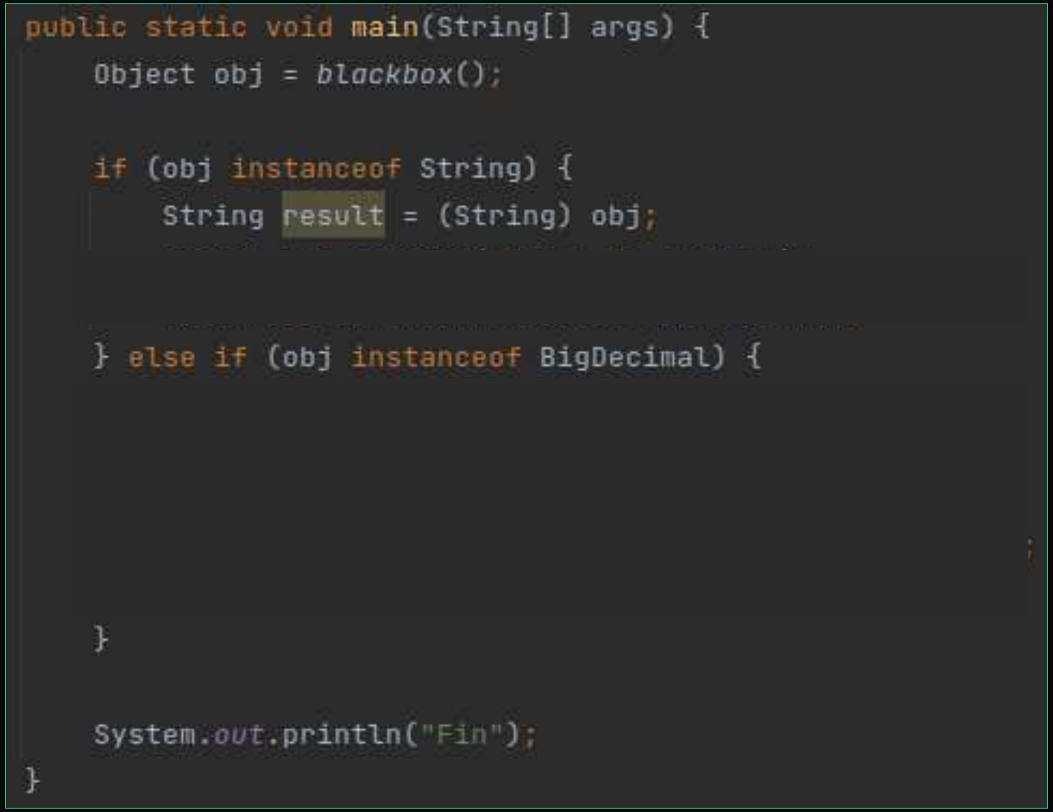
```
/**
 *  simulates blackbox - we don't know what exactly will come out
 *
 * @return either String or BigDecimal
 */
private static Object blackbox() {
```



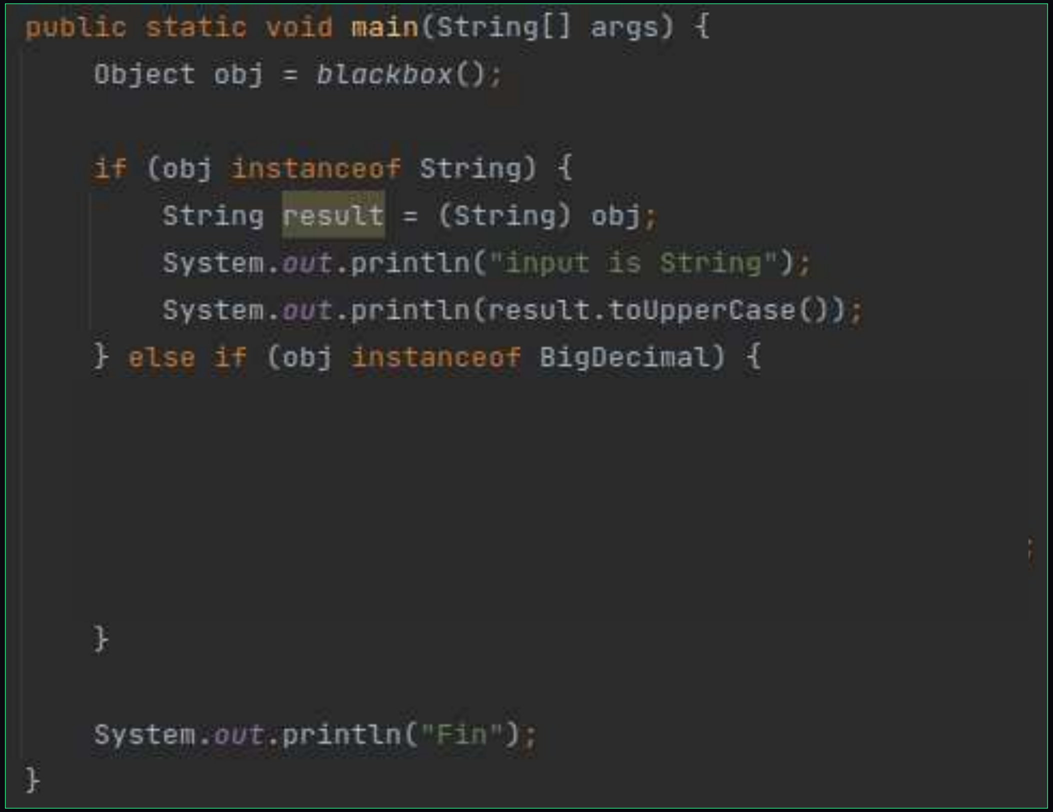
```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String) {









    }

    System.out.println("Fin");
}
```



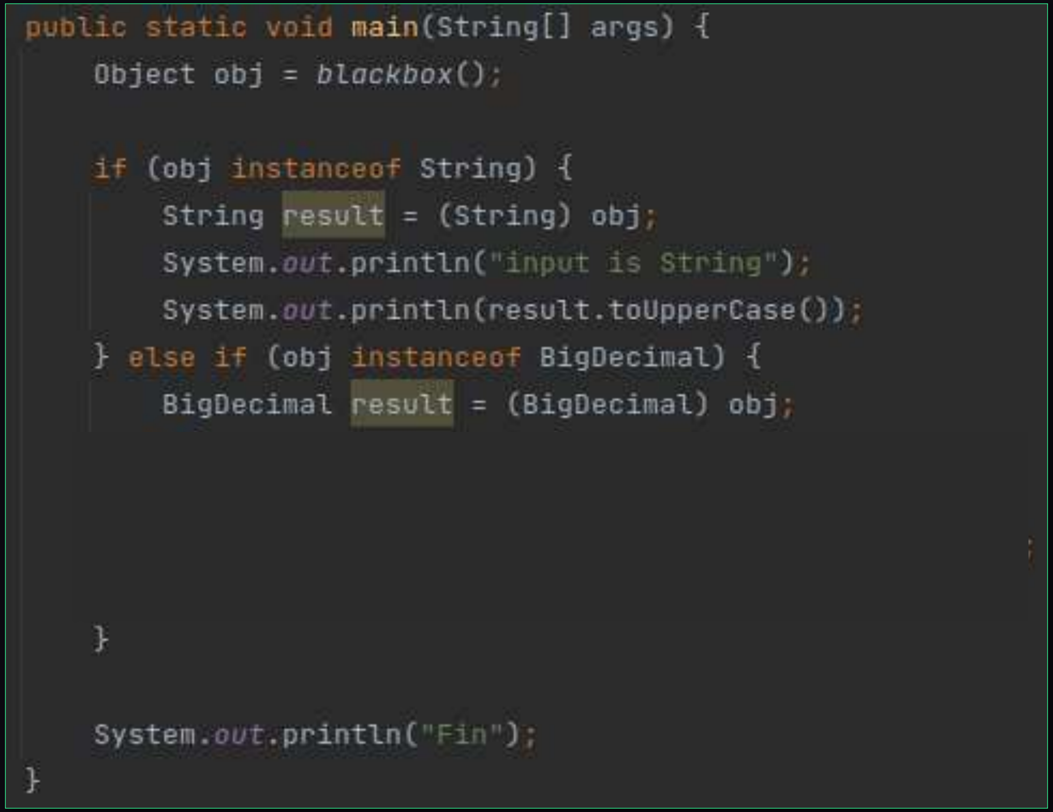
```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String) {



    } else if (obj instanceof BigDecimal) {





    }

    System.out.println("Fin");
}
```



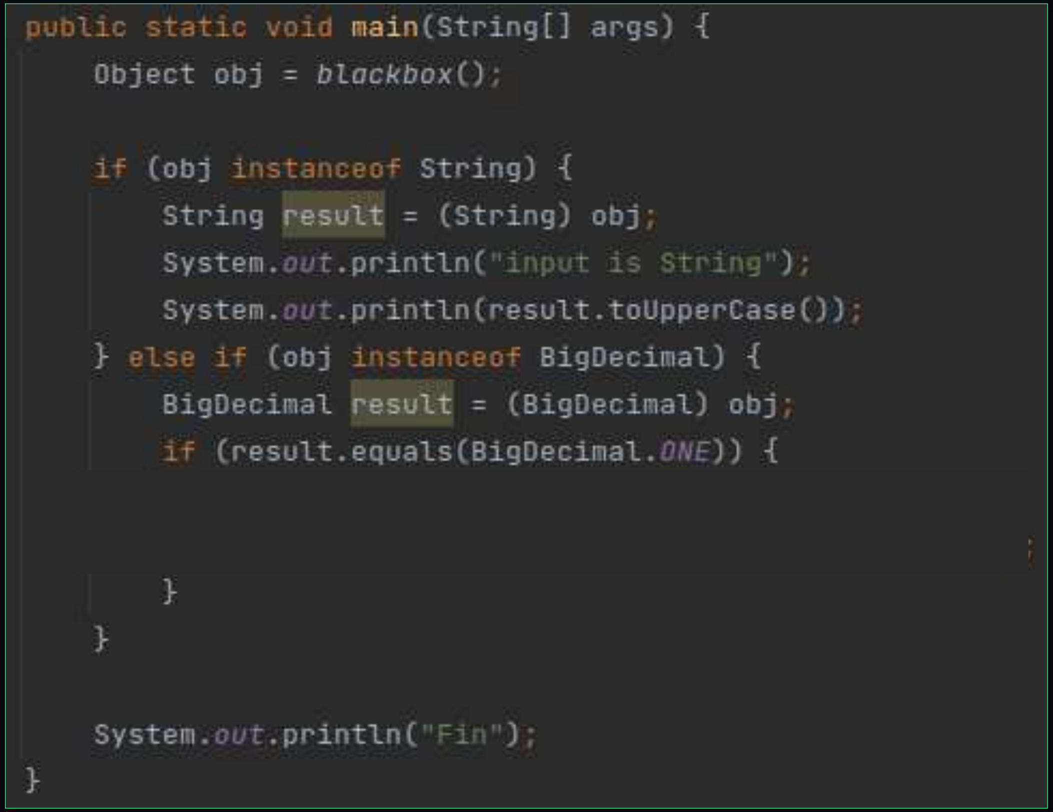
```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String) {
        String result = (String) obj;


    } else if (obj instanceof BigDecimal) {






    }

    System.out.println("Fin");
}
```



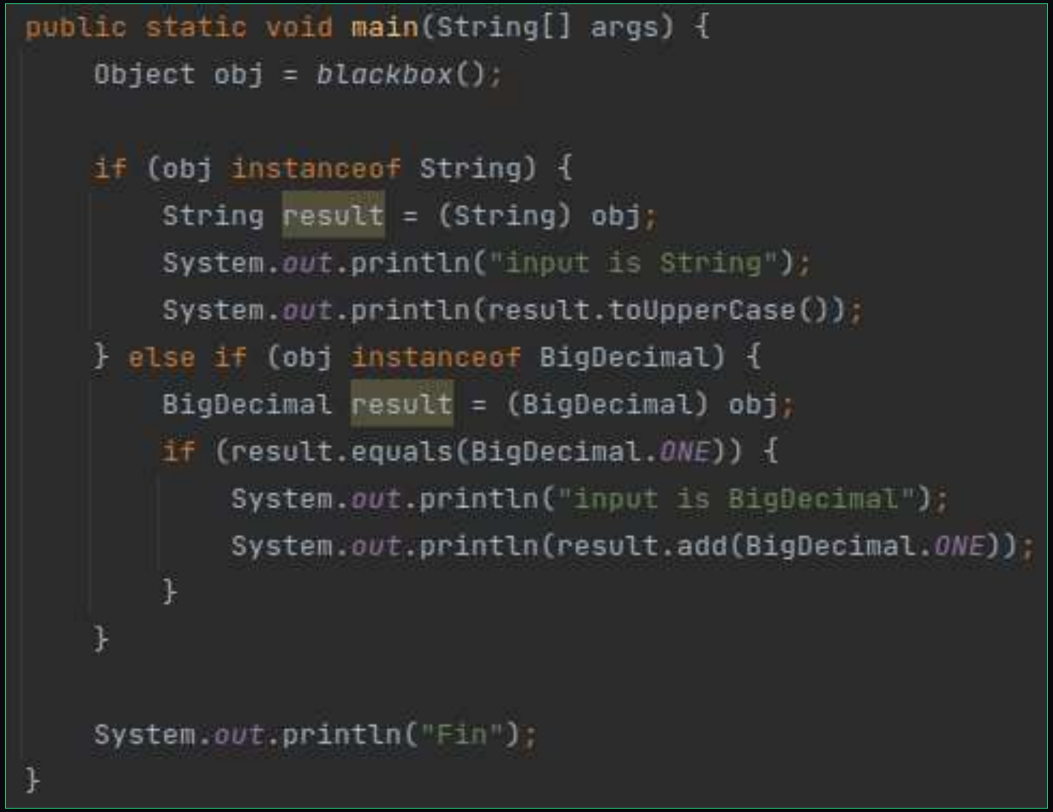
```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String) {
        String result = (String) obj;
        System.out.println("input is String");
        System.out.println(result.toUpperCase());
    } else if (obj instanceof BigDecimal) {





    }

    System.out.println("Fin");
}
```



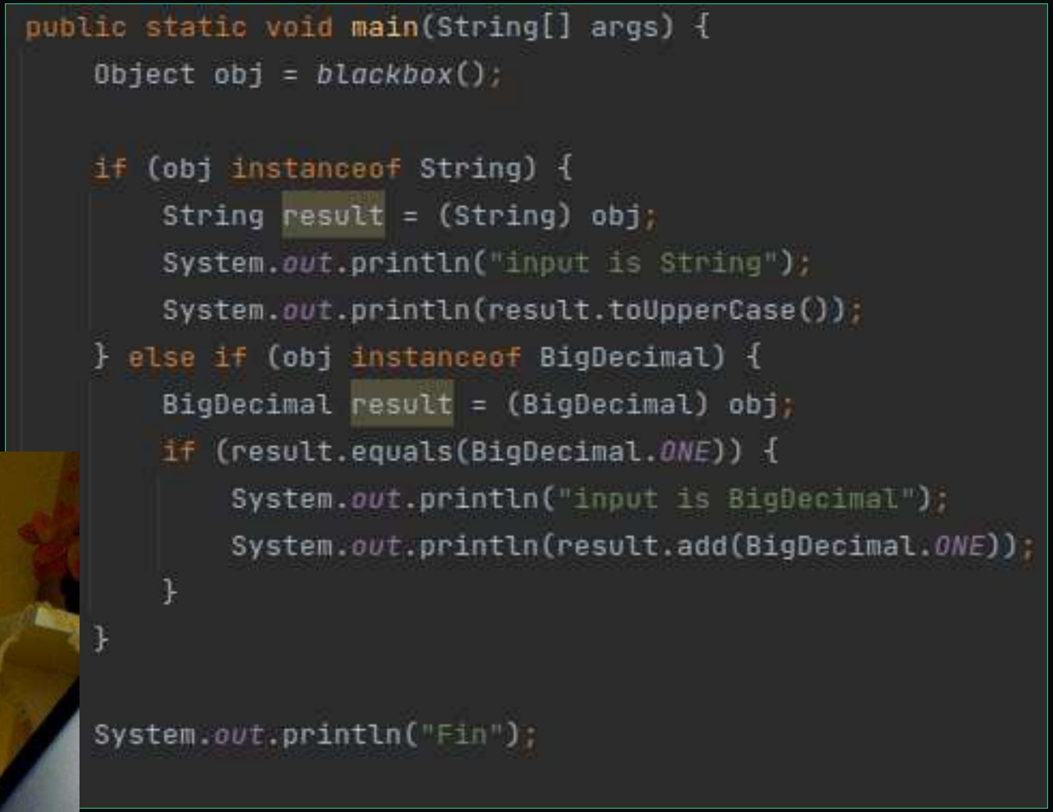
```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String) {
        String result = (String) obj;
        System.out.println("input is String");
        System.out.println(result.toUpperCase());
    } else if (obj instanceof BigDecimal) {
        BigDecimal result = (BigDecimal) obj;




    }

    System.out.println("Fin");
}
```



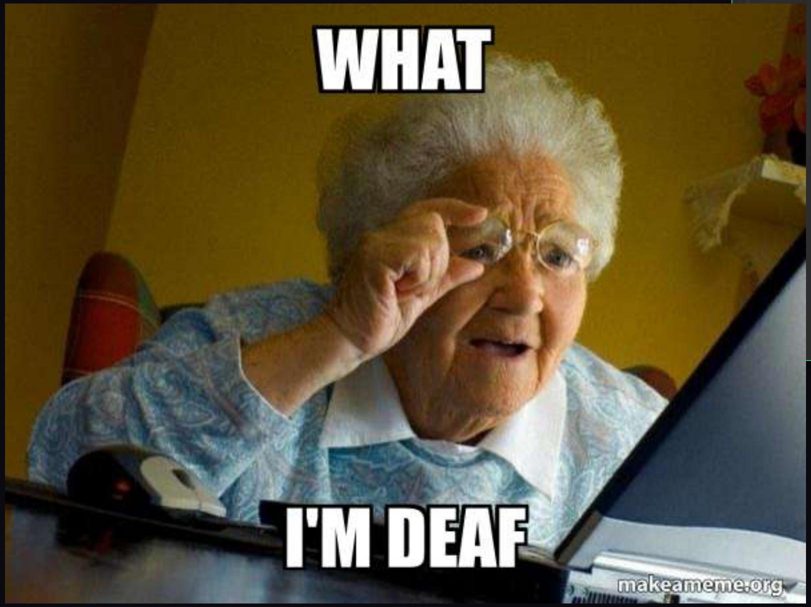
```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String) {
        String result = (String) obj;
        System.out.println("input is String");
        System.out.println(result.toUpperCase());
    } else if (obj instanceof BigDecimal) {
        BigDecimal result = (BigDecimal) obj;
        if (result.equals(BigDecimal.ONE)) {


        }
    }

    System.out.println("Fin");
}
```



```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String) {
        String result = (String) obj;
        System.out.println("input is String");
        System.out.println(result.toUpperCase());
    } else if (obj instanceof BigDecimal) {
        BigDecimal result = (BigDecimal) obj;
        if (result.equals(BigDecimal.ONE)) {
            System.out.println("input is BigDecimal");
            System.out.println(result.add(BigDecimal.ONE));
        }
    }

    System.out.println("Fin");
}
```



```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String) {
        String result = (String) obj;
        System.out.println("input is String");
        System.out.println(result.toUpperCase());
    } else if (obj instanceof BigDecimal) {
        BigDecimal result = (BigDecimal) obj;
        if (result.equals(BigDecimal.ONE)) {
            System.out.println("input is BigDecimal");
            System.out.println(result.add(BigDecimal.ONE));
        }
    }

    System.out.println("Fin");
```

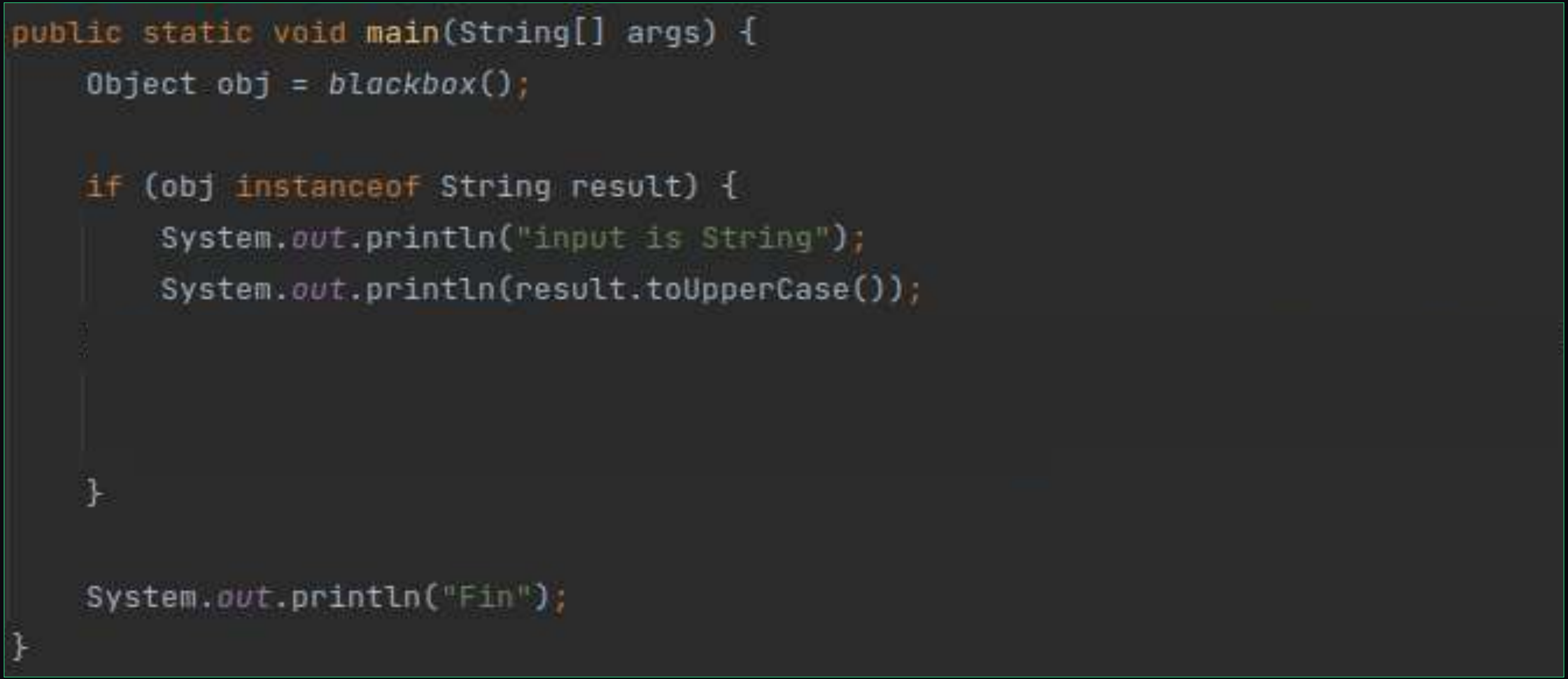


instanceof
without casting?

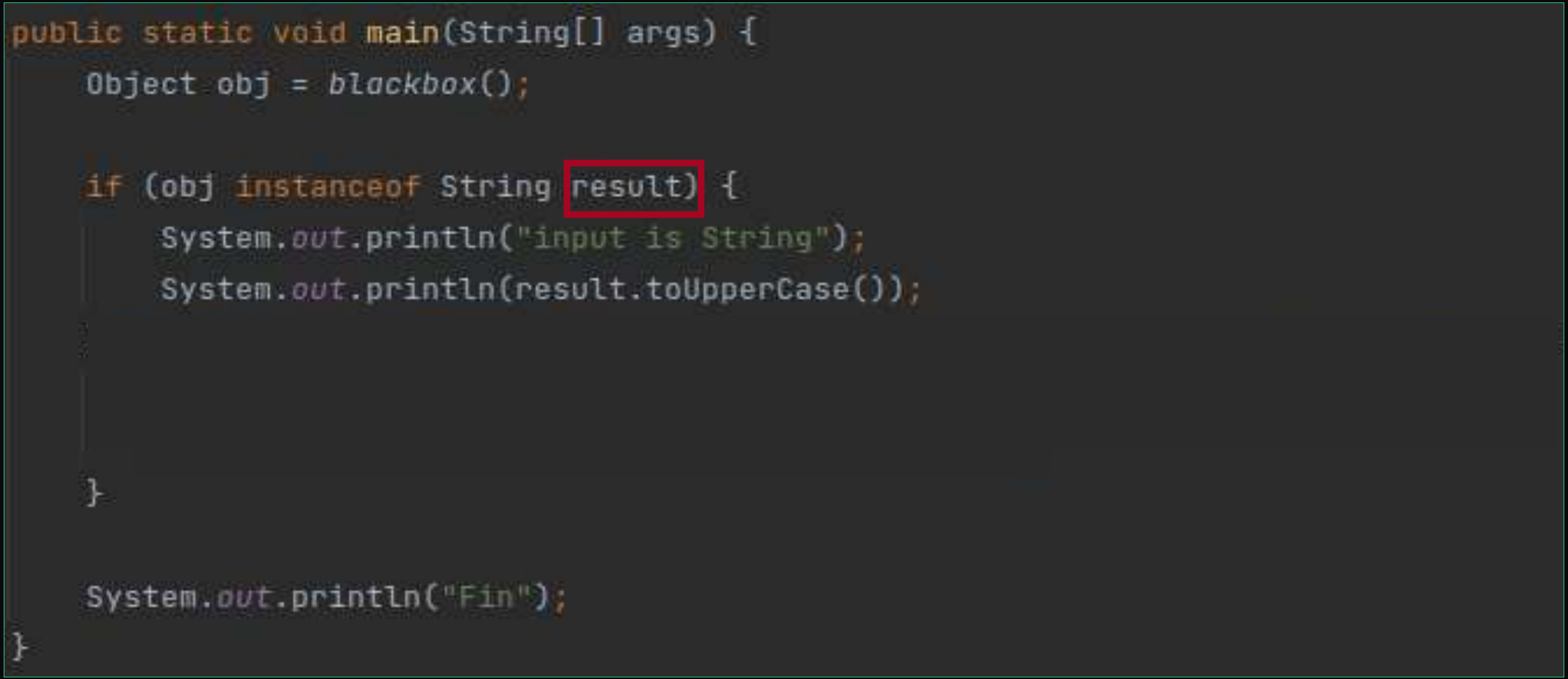
PATTERN MATCHING

Image: deviantart.com

JDK 16

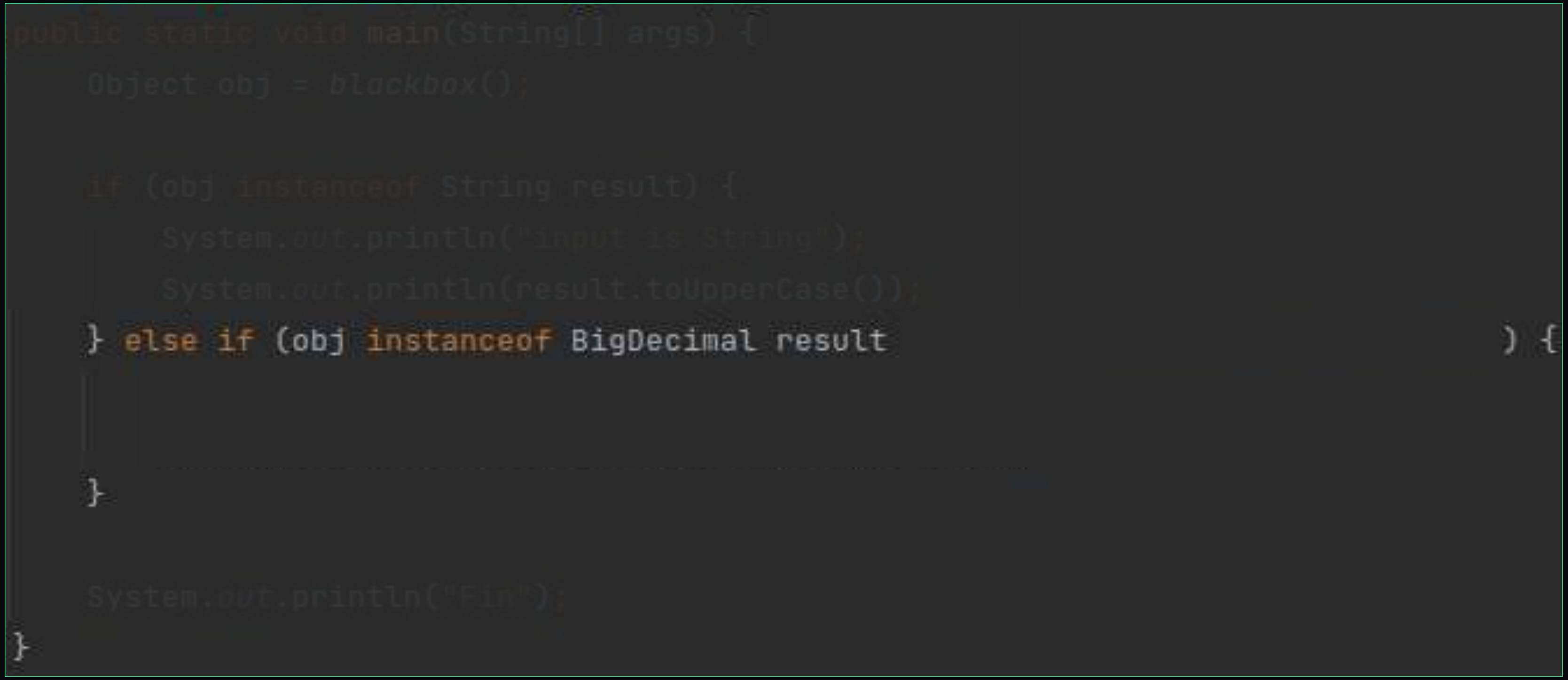


```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String result) {
        System.out.println("input is String");
        System.out.println(result.toUpperCase());


    }

    System.out.println("Fin");
}
```



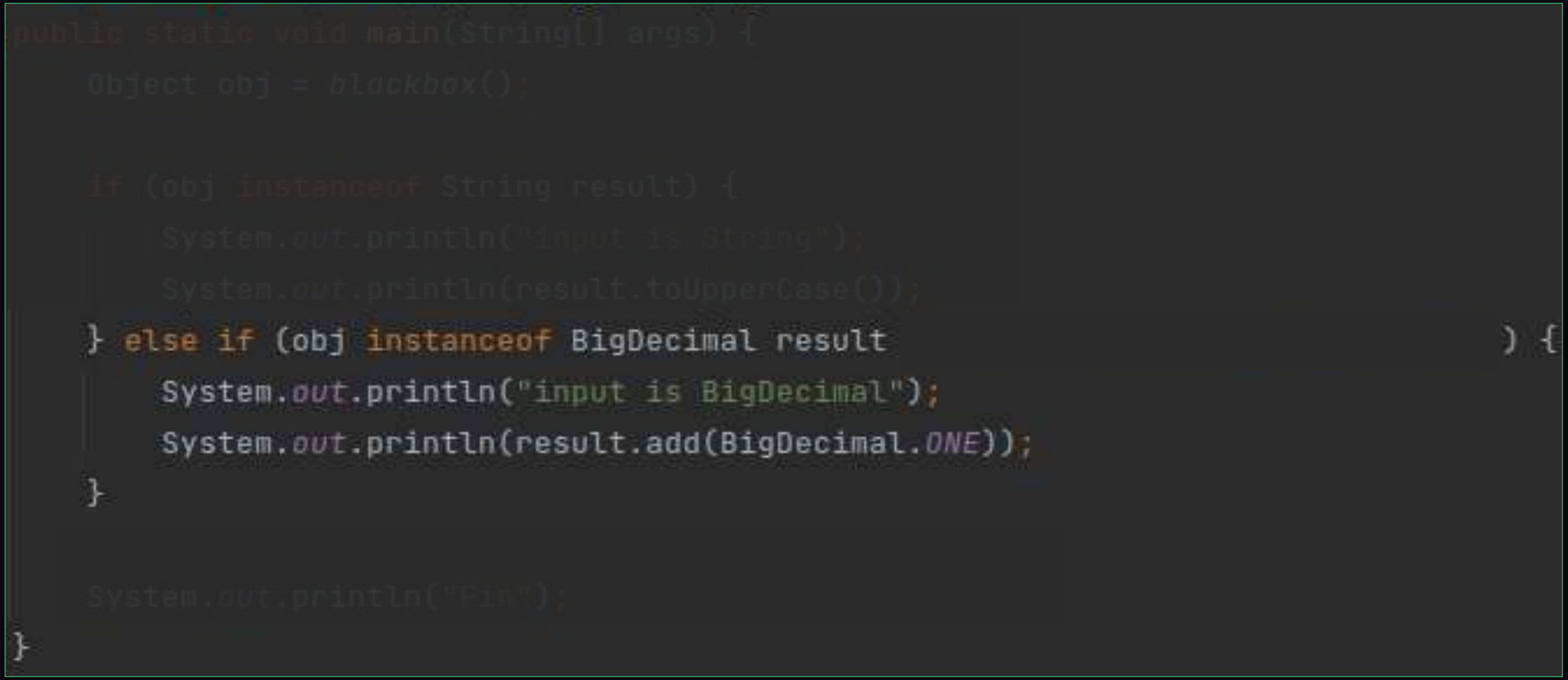
```
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String result) {
        System.out.println("input is String");
        System.out.println(result.toUpperCase());


    }

    System.out.println("Fin");
}
```

JDK 16



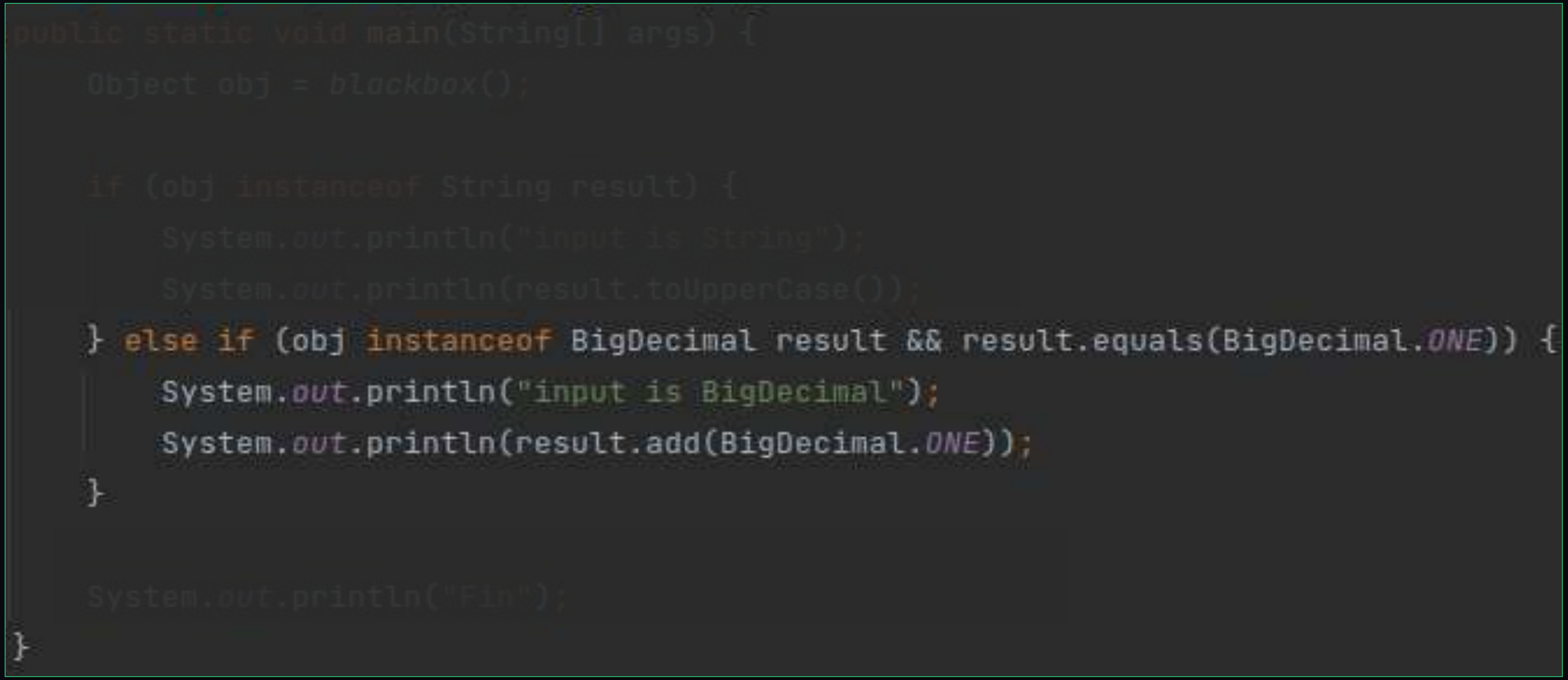
```
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String result) {
        System.out.println("input is String");
        System.out.println(result.toUpperCase());
    } else if (obj instanceof BigDecimal result                                ) {

    }

    System.out.println("Fin");
}
```

JDK 16



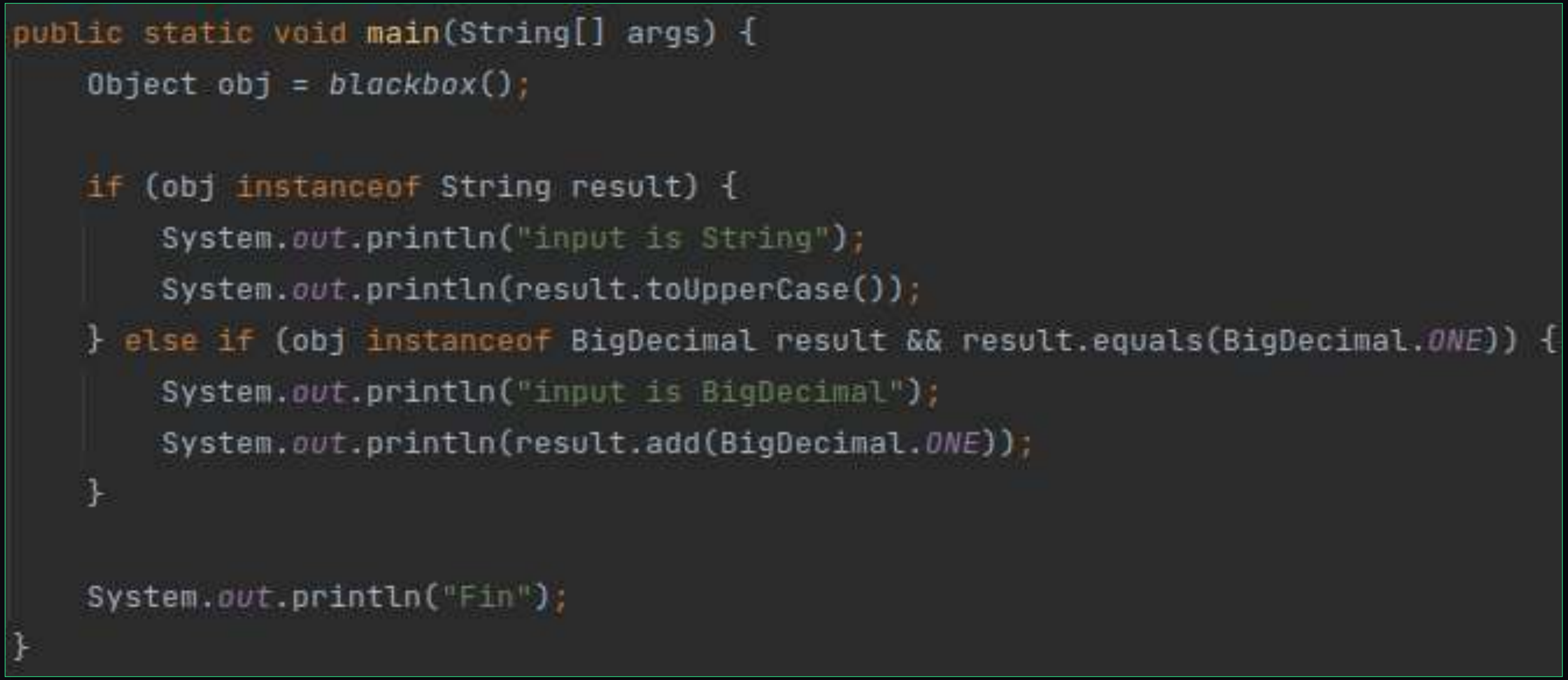
```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String result) {
        System.out.println("input is String");
        System.out.println(result.toUpperCase());
    } else if (obj instanceof BigDecimal result                              ) {
        System.out.println("input is BigDecimal");
        System.out.println(result.add(BigDecimal.ONE));
    }

    System.out.println("Fin");
}
```

JDK 16



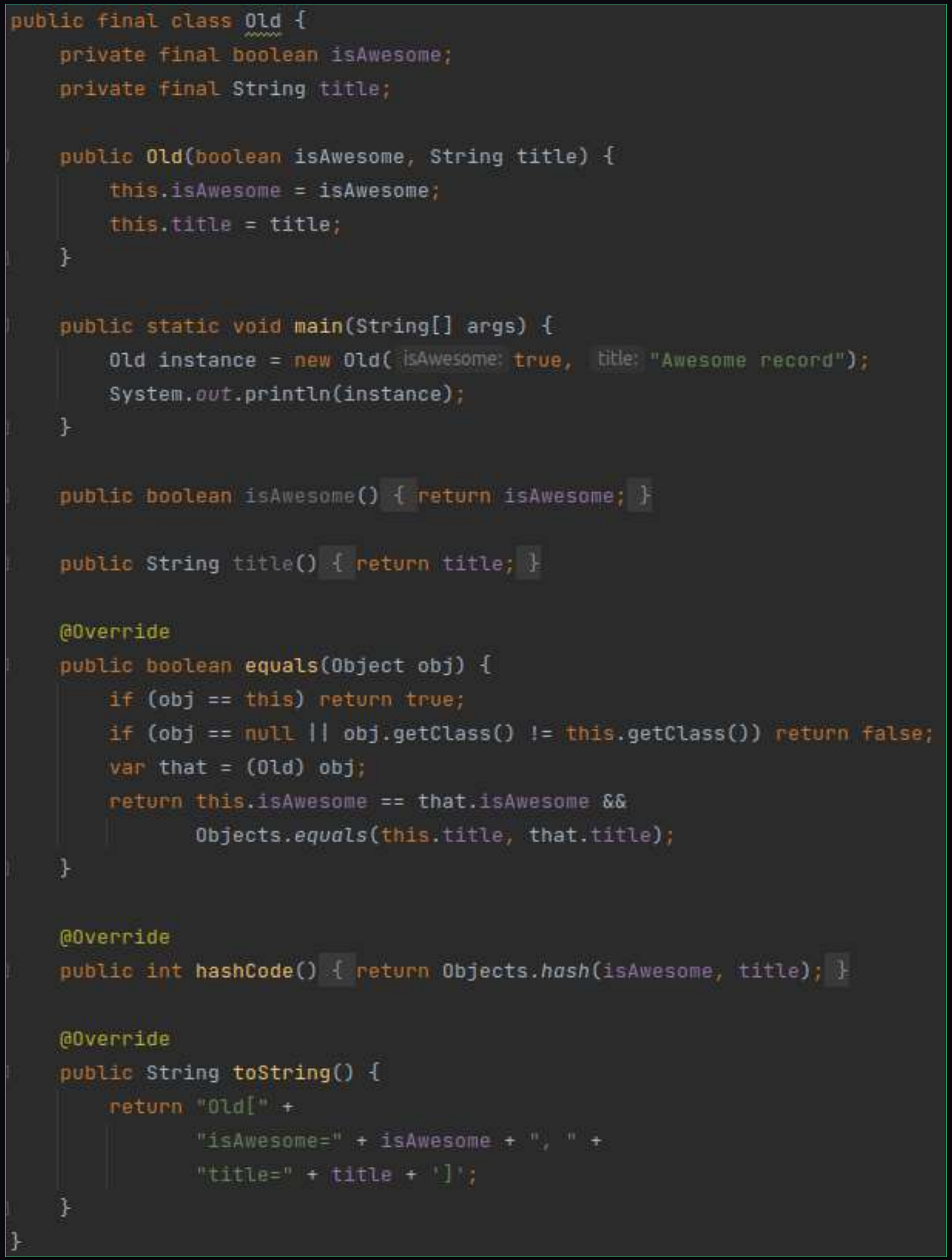
```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String result) {
        System.out.println("input is String");
        System.out.println(result.toUpperCase());
    } else if (obj instanceof BigDecimal result && result.equals(BigDecimal.ONE)) {
        System.out.println("input is BigDecimal");
        System.out.println(result.add(BigDecimal.ONE));
    }

    System.out.println("Fin");
}
```

JDK 16

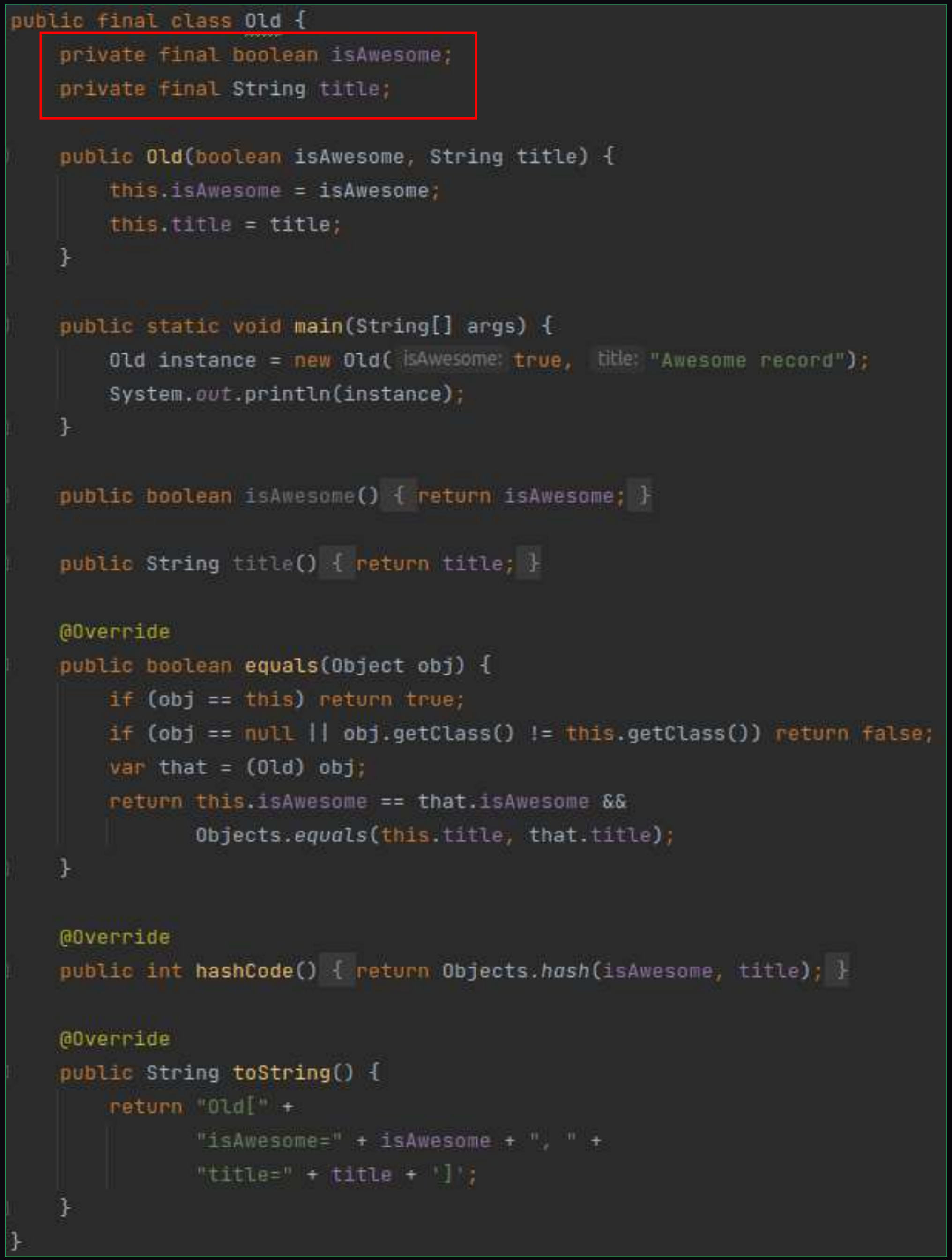


```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof String result) {
        System.out.println("input is String");
        System.out.println(result.toUpperCase());
    } else if (obj instanceof BigDecimal result && result.equals(BigDecimal.ONE)) {
        System.out.println("input is BigDecimal");
        System.out.println(result.add(BigDecimal.ONE));
    }

    System.out.println("Fin");
}
```

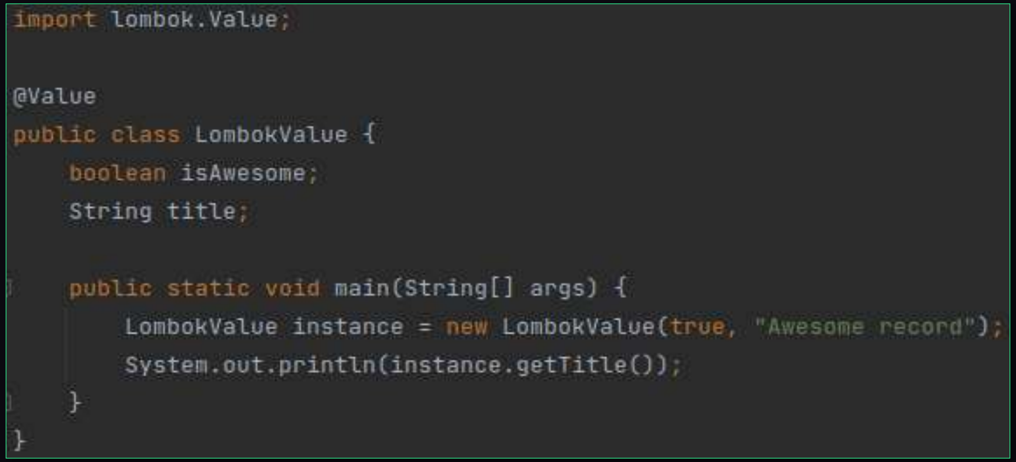
Data class



```java
public final class Old {
    private final boolean isAwesome;
    private final String title;

    public Old(boolean isAwesome, String title) {
        this.isAwesome = isAwesome;
        this.title = title;
    }

    public static void main(String[] args) {
        Old instance = new Old( isAwesome: true,  title: "Awesome record");
        System.out.println(instance);
    }

    public boolean isAwesome() { return isAwesome; }

    public String title() { return title; }

    @Override
    public boolean equals(Object obj) {
        if (obj == this) return true;
        if (obj == null || obj.getClass() != this.getClass()) return false;
        var that = (Old) obj;
        return this.isAwesome == that.isAwesome &&
                Objects.equals(this.title, that.title);
    }

    @Override
    public int hashCode() { return Objects.hash(isAwesome, title); }

    @Override
    public String toString() {
        return "Old[" +
                "isAwesome=" + isAwesome + ", " +
                "title=" + title + ']';
    }
}
```



```java
public final class Old {
    private final boolean isAwesome;
    private final String title;

    public Old(boolean isAwesome, String title) {
        this.isAwesome = isAwesome;
        this.title = title;
    }

    public static void main(String[] args) {
        Old instance = new Old( isAwesome: true,  title: "Awesome record");
        System.out.println(instance);
    }

    public boolean isAwesome() { return isAwesome; }

    public String title() { return title; }

    @Override
    public boolean equals(Object obj) {
        if (obj == this) return true;
        if (obj == null || obj.getClass() != this.getClass()) return false;
        var that = (Old) obj;
        return this.isAwesome == that.isAwesome &&
                Objects.equals(this.title, that.title);
    }

    @Override
    public int hashCode() { return Objects.hash(isAwesome, title); }

    @Override
    public String toString() {
        return "Old[" +
                "isAwesome=" + isAwesome + ", " +
                "title=" + title + ']';
    }
}
```



```java
import lombok.Value;

@Value
public class LombokValue {
    boolean isAwesome;
    String title;

    public static void main(String[] args) {
        LombokValue instance = new LombokValue(true, "Awesome record");
        System.out.println(instance.getTitle());
    }
}
```

We're approaching record

RECORD

Image: deviantart.com

```java
public record New(boolean isAwesome, String title) {

  public static void main(String[] args) {
    New instance = new New(true, "Awesome record");
    System.out.println(instance.title());
  }
}
```

What’s the record in the number of children born?

Fiodorowa Wasiljewa

69

What does that have to do with Java?

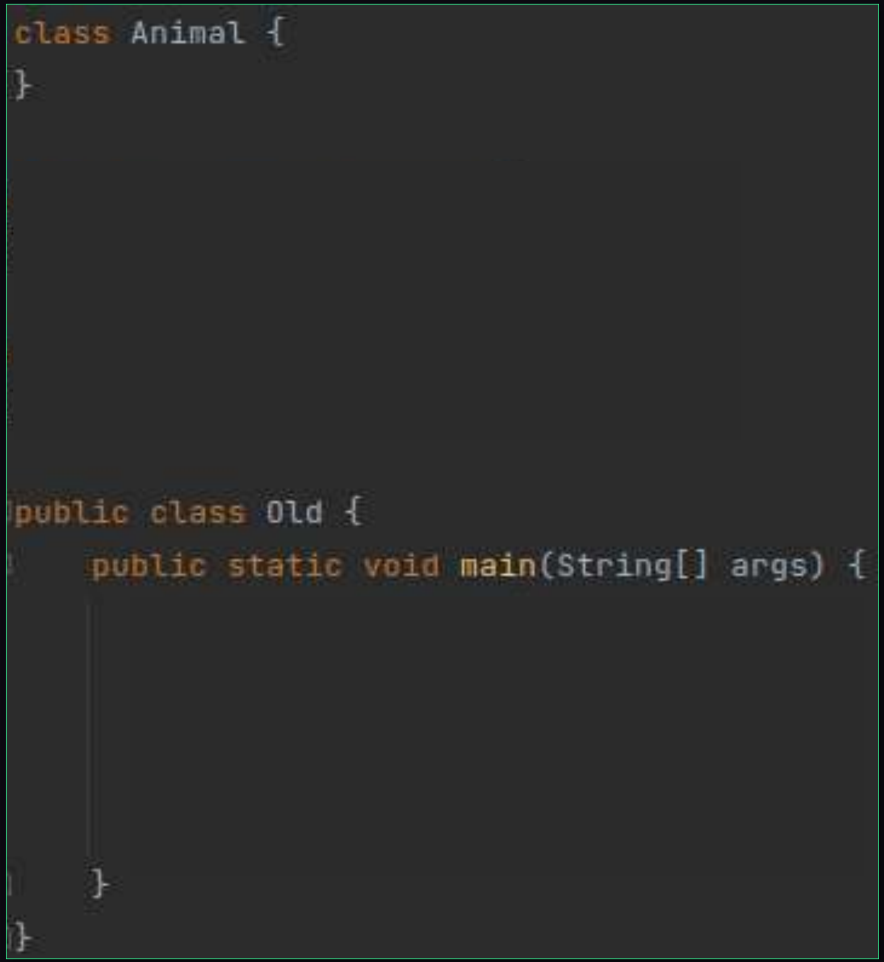
Inheritance

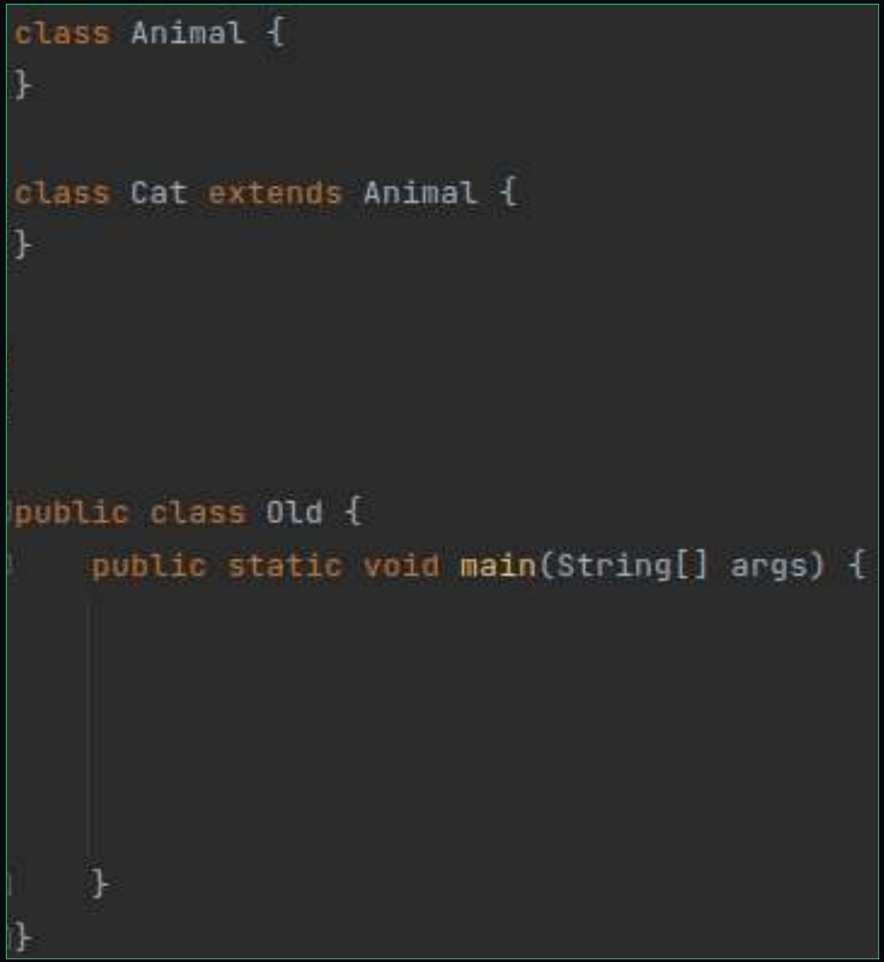
final

SEALED CLASSES

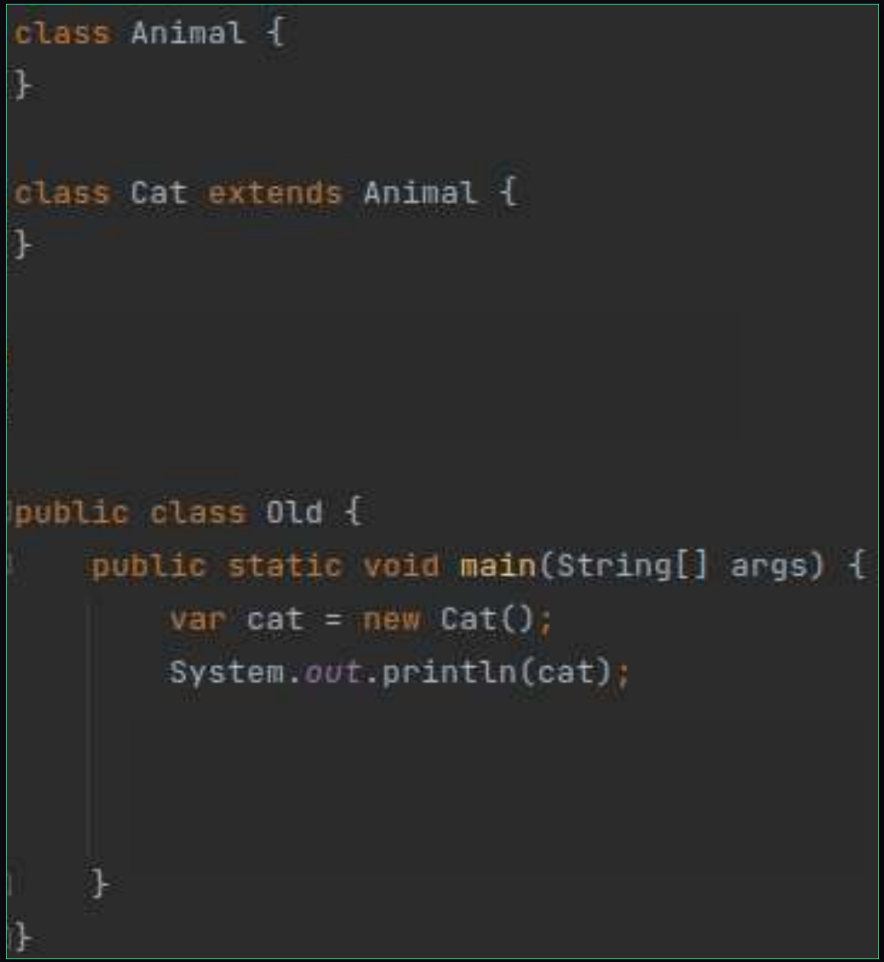
Image: deviantart.com

Contraception for Java

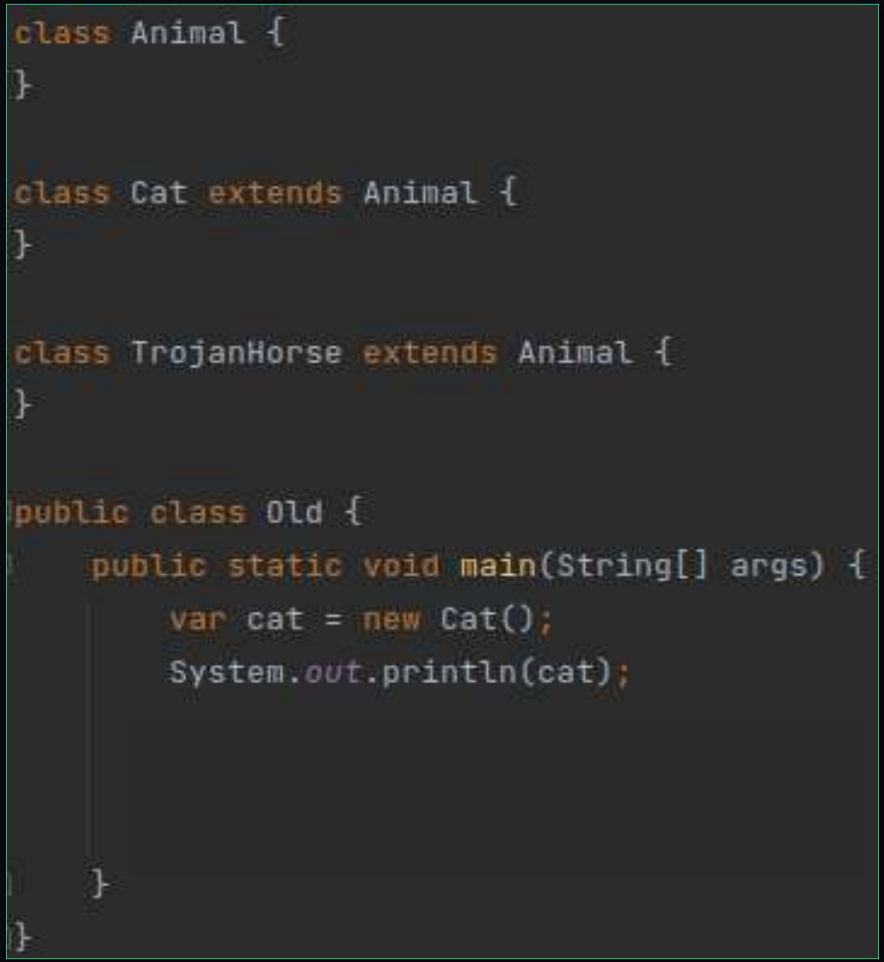

```java
class Animal {
}







public class Old {
    public static void main(String[] args) {




    }
}
```



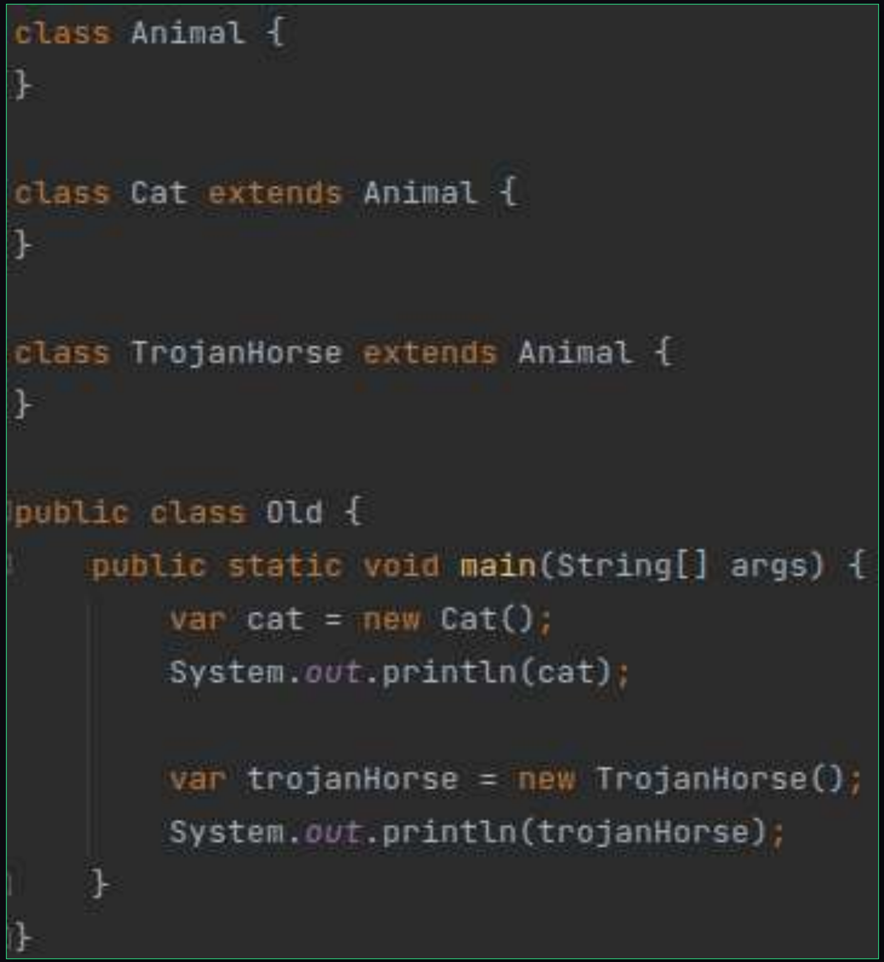
```java
class Animal {
}

class Cat extends Animal {
}




public class Old {
    public static void main(String[] args) {




    }
}
```



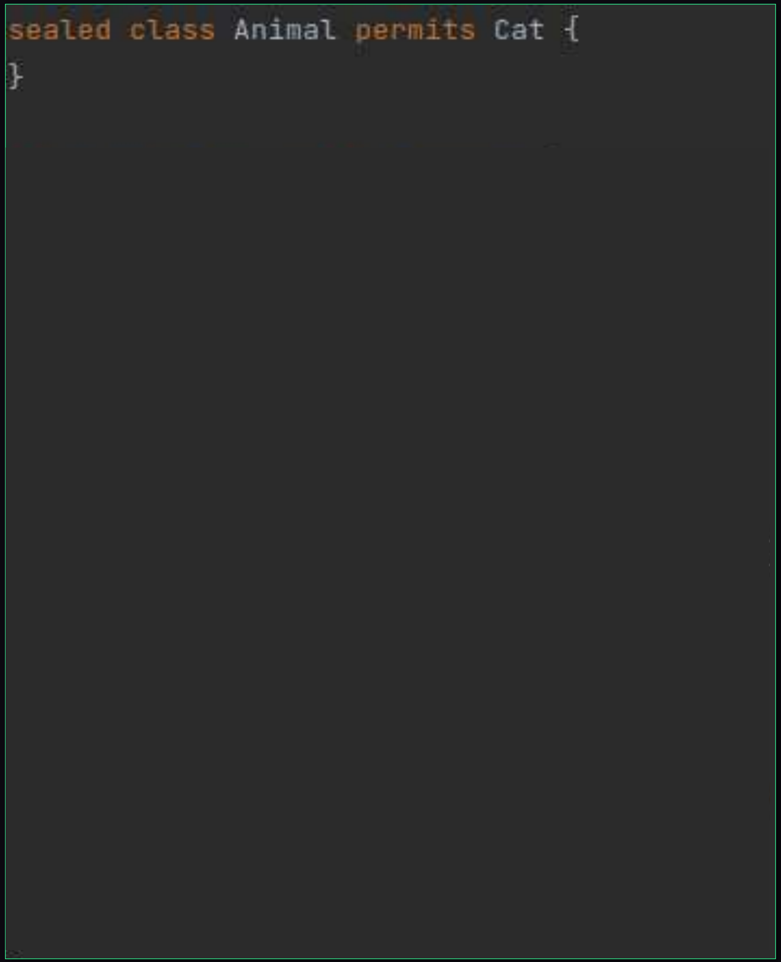
```java
class Animal {
}

class Cat extends Animal {
}



public class Old {
    public static void main(String[] args) {
        var cat = new Cat();
        System.out.println(cat);


    }
}
```



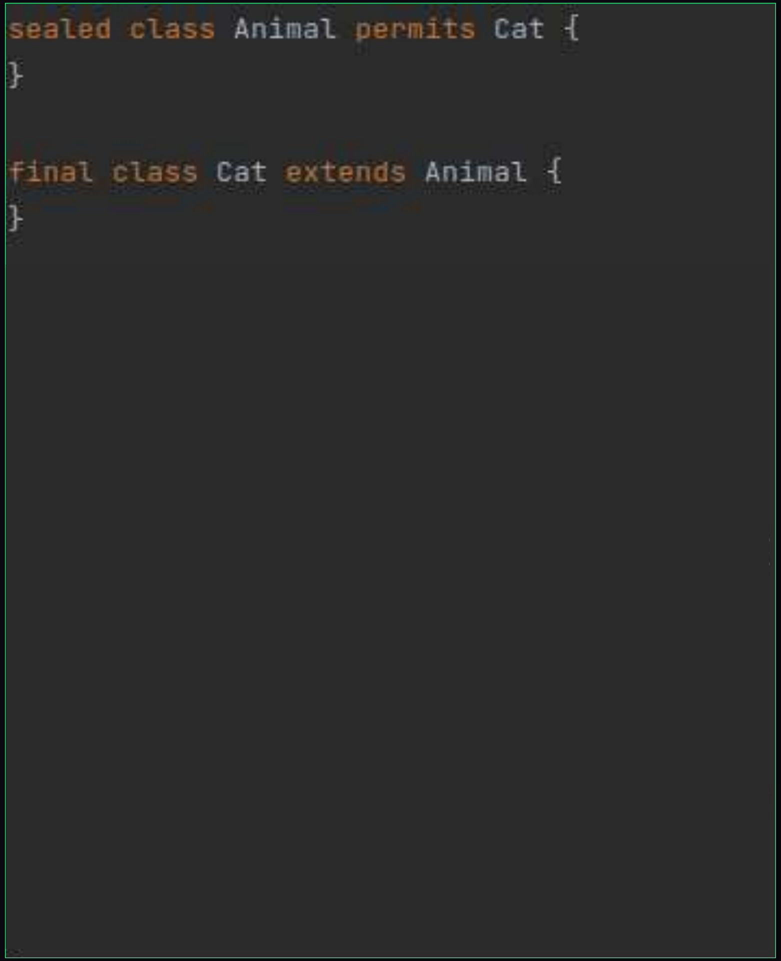
```java
class Animal {
}

class Cat extends Animal {
}

class TrojanHorse extends Animal {
}

public class Old {
    public static void main(String[] args) {
        var cat = new Cat();
        System.out.println(cat);



    }
}
```



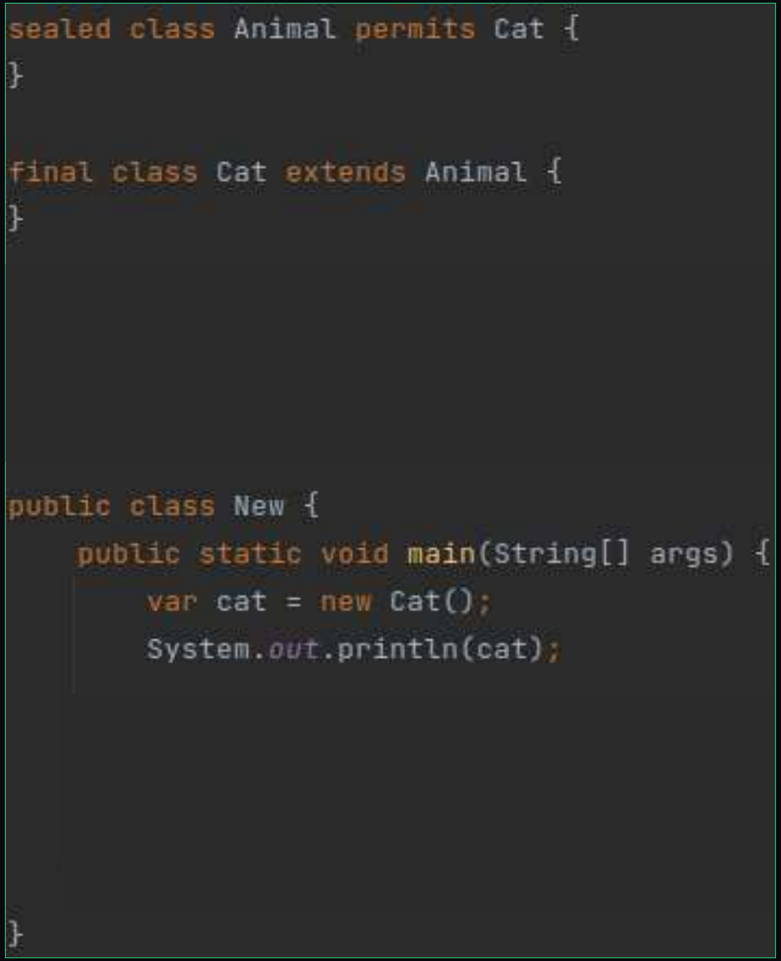
```java
class Animal {
}

class Cat extends Animal {
}

class TrojanHorse extends Animal {
}

public class Old {
    public static void main(String[] args) {
        var cat = new Cat();
        System.out.println(cat);

        var trojanHorse = new TrojanHorse();
        System.out.println(trojanHorse);
    }
}
```



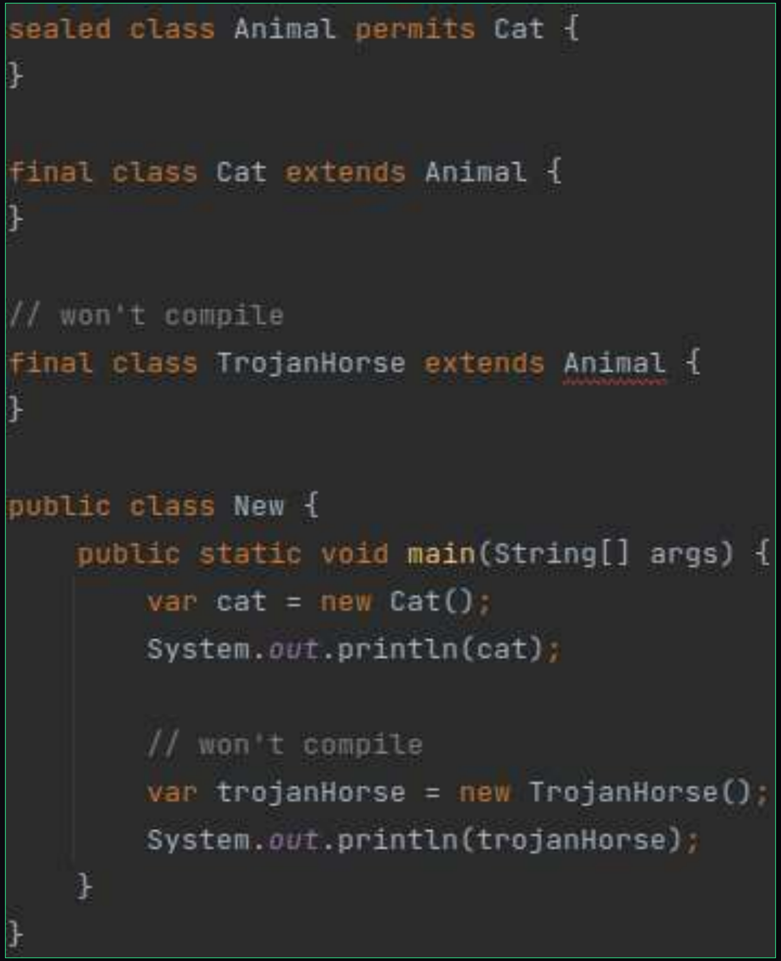
```
sealed class Animal permits Cat {
}
```



```java
sealed class Animal permits Cat {
}

final class Cat extends Animal {
}
```



```java
sealed class Animal permits Cat {
}

final class Cat extends Animal {
}





public class New {
    public static void main(String[] args) {
        var cat = new Cat();
        System.out.println(cat);




}
```



```java
sealed class Animal permits Cat {
}

final class Cat extends Animal {
}

// won't compile
final class TrojanHorse extends Animal {
}

public class New {
    public static void main(String[] args) {
        var cat = new Cat();
        System.out.println(cat);

        // won't compile
        var trojanHorse = new TrojanHorse();
        System.out.println(trojanHorse);
    }
}
```

sealed non-sealed permits

```
sealed class Animal permits Cat {}

non-sealed class Cat extends Animal {}

class OutdoorCat extends Cat {}
```

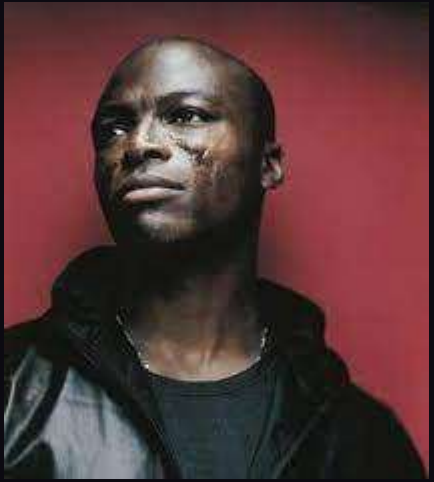
JDK 17

```java
sealed class Animal permits Cat {}

non-sealed class Cat extends Animal {}

class OutdoorCat extends Cat {}

final class DomesticCat extends Cat {}
```

What kind of music do Java's classes listen to?

final class

sealed class



non-sealed class

class

```
class Parent { }
```

class

```
class Parent { }

class UnexpectedChild extends Parent {}
```

But the kid is not my son…

FINALIZERS

Image: deviantart.com

01 WINS
00 WINS
REPTILE
KUNG LAO
FINISH HIM !!

RECORD PATTERNS

Image: deviantart.com

```java
record Point(int x, int y) { }
```

```java
record Rectangle(Point topLeft, Point bottomRight) {}
```

```java
public static void main(String[] args) {
   Object obj = blackbox();

   if (obj instanceof Rectangle rect) {









   }

   System.out.println("Fin");
}
```

```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof Rectangle rect) {









        System.out.println("Width: " + width);
        System.out.println("Height: " + height);
    }

    System.out.println("Fin");
}
```

```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof Rectangle rect) {
        var topLeft = rect.topLeft();
        var bottomRight = rect.bottomRight();







        System.out.println("Width: " + width);
        System.out.println("Height: " + height);
    }

    System.out.println("Fin");
}
```

```
public static void main(String[] args) {
   Object obj = blackbox();

   if (obj instanceof Rectangle rect) {
      var topLeft = rect.topLeft();
      var bottomRight = rect.bottomRight();
      var left = topLeft.x();
      var right = bottomRight.x();





      System.out.println("Width: " + width);
      System.out.println("Height: " + height);
   }

   System.out.println("Fin");
}
```

```java
public static void main(String[] args) {
   Object obj = blackbox();

   if (obj instanceof Rectangle rect) {
      var topLeft = rect.topLeft();
      var bottomRight = rect.bottomRight();
      var left = topLeft.x();
      var right = bottomRight.x();
      var width = right - left;




      System.out.println("Width: " + width);
      System.out.println("Height: " + height);
   }

   System.out.println("Fin");
}
```

```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof Rectangle rect) {
        var topLeft = rect.topLeft();
        var bottomRight = rect.bottomRight();
        var left = topLeft.x();
        var right = bottomRight.x();
        var width = right - left;

        var top = topLeft.y();
        var bottom = bottomRight.y();
        var height = bottom - top;
        System.out.println("Width: " + width);
        System.out.println("Height: " + height);
    }

    System.out.println("Fin");
}
```

```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof Rectangle rect) {
        var topLeft = rect.topLeft();
        var bottomRight = rect.bottomRight();
        var left = topLeft.x();
        var right = bottomRight.x();
        var width = right - left;
        var top = topLeft.y();
        var bottom = bottomRight.y();
        var height = bottom - top;
        System.out.println("Width: " + width);
        System.out.println("Height: " + height);
    }

    System.out.println("Fin");
}
```

10 lines

What about squizing to 4 lines?

```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof Rectangle                                                              ) {


        System.out.println("Width: " + width);
        System.out.println("Height: " + height);
    }

    System.out.println("Fin");
}
```

```java
record Point(int x, int y) { }
```

```java
record Rectangle(Point topLeft, Point bottomRight) {}
```

```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof Rectangle                                   ) {


        System.out.println("Width: " + width);
        System.out.println("Height: " + height);
    }

    System.out.println("Fin");
}
```

```java
record Point(int x, int y) { }
```

```java
record Rectangle(Point topLeft, Point bottomRight) {}
```

```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof Rectangle                                                              ) {
        var width = rightX - leftX;
        var height = bottomY - topY;
        System.out.println("Width: " + width);
        System.out.println("Height: " + height);
    }

    System.out.println("Fin");
}
```

```java
record Point(int x, int y) { }
```

```java
record Rectangle(Point topLeft, Point bottomRight) {}
```

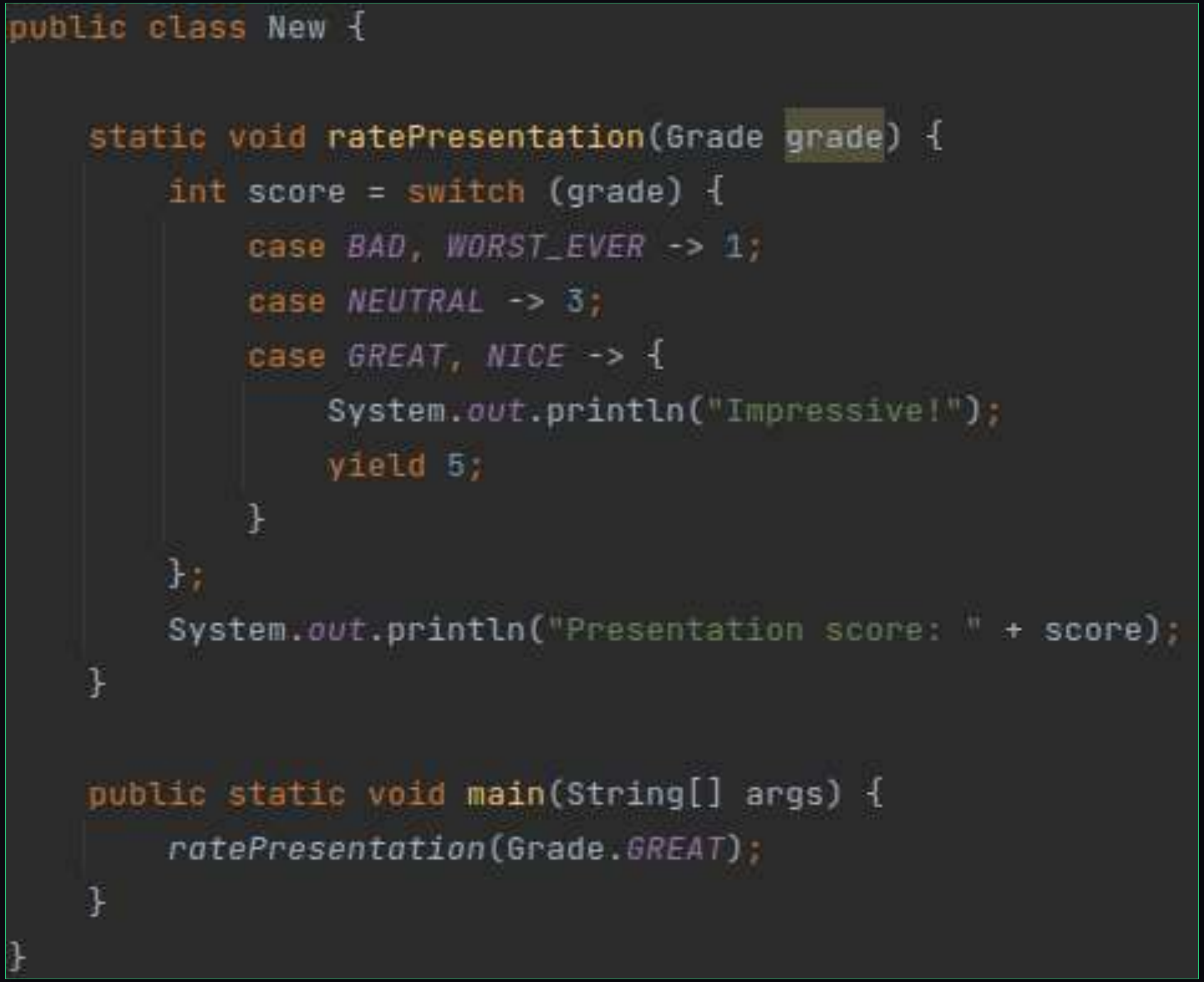
```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof Rectangle(Point(int leftX, int topY), Point(int rightX, int bottomY))) {
        var width = rightX - leftX;
        var height = bottomY - topY;
        System.out.println("Width: " + width);
        System.out.println("Height: " + height);
    }

    System.out.println("Fin");
}
```

```java
record Point(int x, int y) { }
```

```java
record Rectangle(Point topLeft, Point bottomRight) {}
```

```java
public static void main(String[] args) {
    Object obj = blackbox();

    if (obj instanceof Rectangle(Point(int leftX, int topY), Point(int rightX, int bottomY))) {
        var width = rightX - leftX;
        var height = bottomY - topY;
        System.out.println("Width: " + width);
        System.out.println("Height: " + height);
    }

    System.out.println("Fin");
}
```

SWITCH ENHACEMENTS



```java
public class New {

    static void ratePresentation(Grade grade) {
        int score = switch (grade) {
            case BAD, WORST_EVER -> 1;
            case NEUTRAL -> 3;
            case GREAT, NICE -> {
                System.out.println("Impressive!");
                yield 5;
            }
        };
        System.out.println("Presentation score: " + score);
    }

    public static void main(String[] args) {
        ratePresentation(Grade.GREAT);
    }
}
```

```
public class Old {

  static void ratePresentation(Grade grade) {
    if (grade == null) {
      System.out.println("Unfortunately grade is null");
      return;
    }
    int score = switch (grade) {
      case BAD, WORST_EVER -> 1;
      case NEUTRAL -> 3;
      case GREAT, NICE -> {
        System.out.println("Wow!");
        yield 5;
      }
    };
    System.out.println("Presentation score: " + score);
  }

  public static void main(String[] args) {
    ratePresentation(null);
  }
}
```

```
public class New {

   static void ratePresentation(Grade grade) {
      int score = switch (grade) {
         case null, BAD, WORST_EVER -> 1;
         case NEUTRAL -> 3;
         case GREAT, NICE -> {
            System.out.println("Wow!");
            yield 5;
         }
      };
      System.out.println("Presentation score: " + score);
   }

   public static void main(String[] args) {
      ratePresentation(null);
   }
}
```

SWITCH ENHANCEMENTS

pattern matching

Image: deviantart.com

```java
public static void main(String[] args) {
    Object obj = blackbox();

    switch(obj) {
        case BigDecimal bigDecimal ->
            System.out.println("Big decimal: " + bigDecimal.add(BigDecimal.ONE));




    }
}
```

```
public static void main(String[] args) {
    Object obj = blackbox();

    switch(obj) {
        case BigDecimal bigDecimal ->
             System.out.println("Big decimal: " + bigDecimal.add(BigDecimal.ONE));
        case String str ->
              System.out.println("String: " + str.toUpperCase());



    }
}
```

JDK 17-21

```java
public static void main(String[] args) {
    Object obj = blackbox();

    switch(obj) {
        case BigDecimal bigDecimal ->
             System.out.println("Big decimal: " + bigDecimal.add(BigDecimal.ONE));
        case String str ->
              System.out.println("String: " + str.toUpperCase());
        case null, default ->
               System.out.println("World is unexpected!");
    }
}
```

SWITCH ENHACEMENTS

cases order

```java
static void error(Object o) {
   switch (o) {






   }
}
```

```
static void error(Object o) {
   switch (o) {




      default -> {}
   }
}
```

```
static void error(Object o) {
    switch (o) {
        case CharSequence cs ->
                System.out.println("A sequence of length " + cs.length());


        default -> {}
    }
}
```

```
static void error(Object o) {
   switch (o) {
      case CharSequence cs ->
            System.out.println("A sequence of length " + cs.length());
      case String s ->
            System.out.println("A string: " + s);
      default -> {}
   }
}
```

```java
static void error(Object o) {
   switch (o) {
      case CharSequence cs ->
            System.out.println("A sequence of length " + cs.length());
      case String s ->                                                       // won't compile
            System.out.println("A string: " + s);
      default -> {}
   }
}
```

```
static void error(Object o) {
    switch (o) {
        case String s ->
                System.out.println("A string: " + s);
        case CharSequence cs ->
                System.out.println("A sequence of length " + cs.length());
        default -> {}
    }
}
```

SWITCH ENHACEMENTS

Sealed classes

```
sealed interface Animal permits Cat, Dog, Fish { }

final class Cat implements Animal { }
final class Dog implements Animal { }
final class Fish implements Animal { }
```

```
static void checkAnimal(Animal animal) {
    switch (animal) {



    }
}
```

```
static void checkAnimal(Animal animal) {
    switch (animal) {
        case Cat ignored -> System.out.println("just cat");


    }
}
```

```
static void checkAnimal(Animal animal) {
   switch (animal) {
      case Cat ignored -> System.out.println("just cat");
      case Dog ignored -> System.out.println("just dog");

   }
}
```

```
static void checkAnimal(Animal animal) {
   switch (animal) {
      case Cat ignored -> System.out.println("just cat");
      case Dog ignored -> System.out.println("just dog");
      case Fish ignored ->  System.out.println("just fish");
   }
}
```

```
sealed interface Animal permits Cat, Dog, Fish { }

final class Cat implements Animal { }
final class Dog implements Animal { }
final class Fish implements Animal { }
```

```
static void checkAnimal(Animal animal) {
   switch (animal) {
      case Cat ignored -> System.out.println("just cat");
      case Dog ignored -> System.out.println("just dog");
      case Fish ignored ->  System.out.println("just fish");
   }
}
```

Threads

Thread

# threads in Java

=

# os threads

<=

MAX # os threads

JDK 19-21

VIRTUAL THREADS

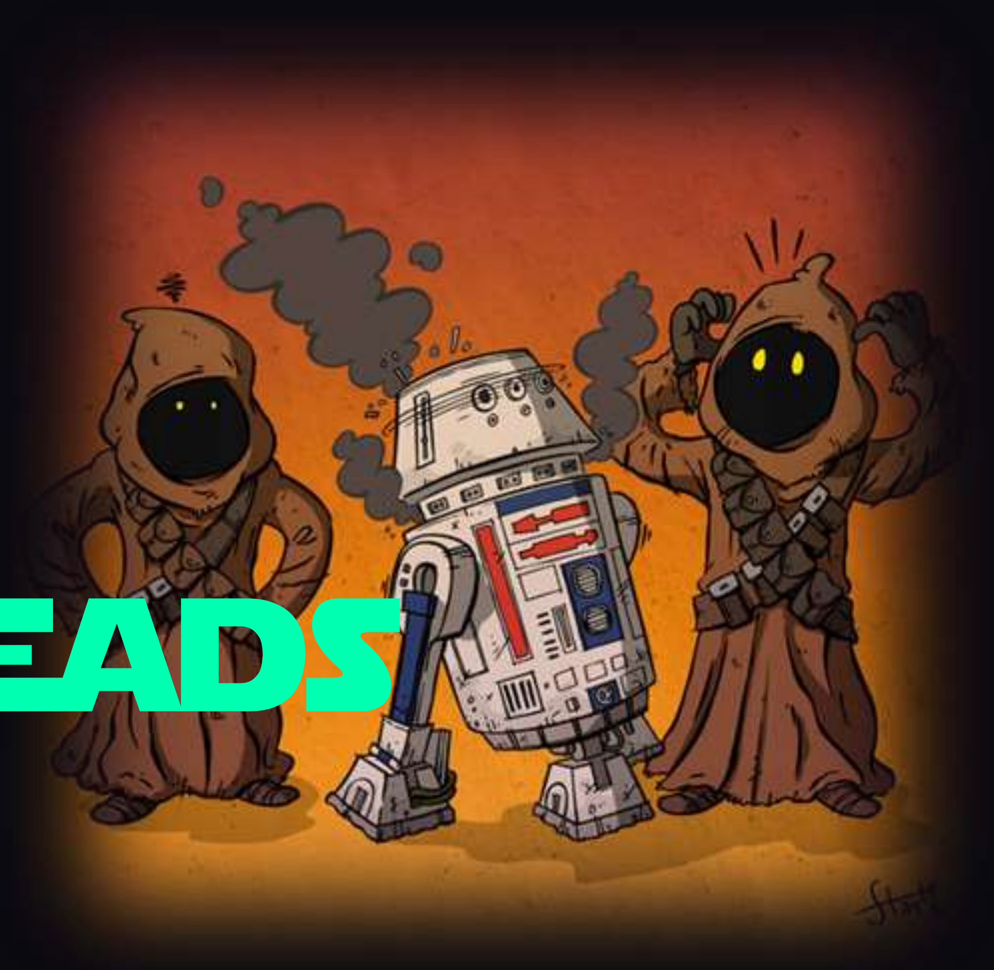


Image: deviantart.com

Virtual thread

# threads in Java

>

MAX # os threads

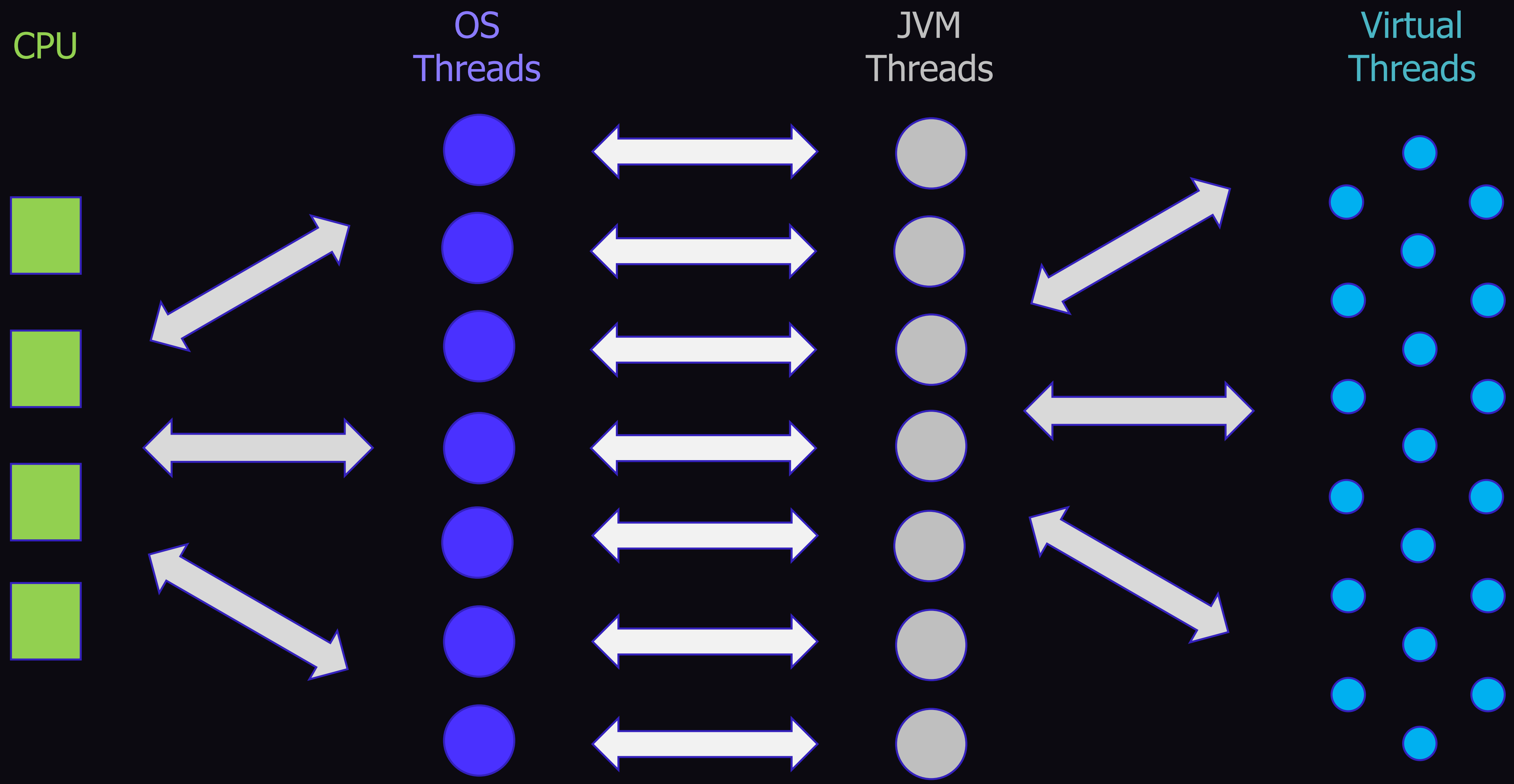
Virtual thread

# threads in Java

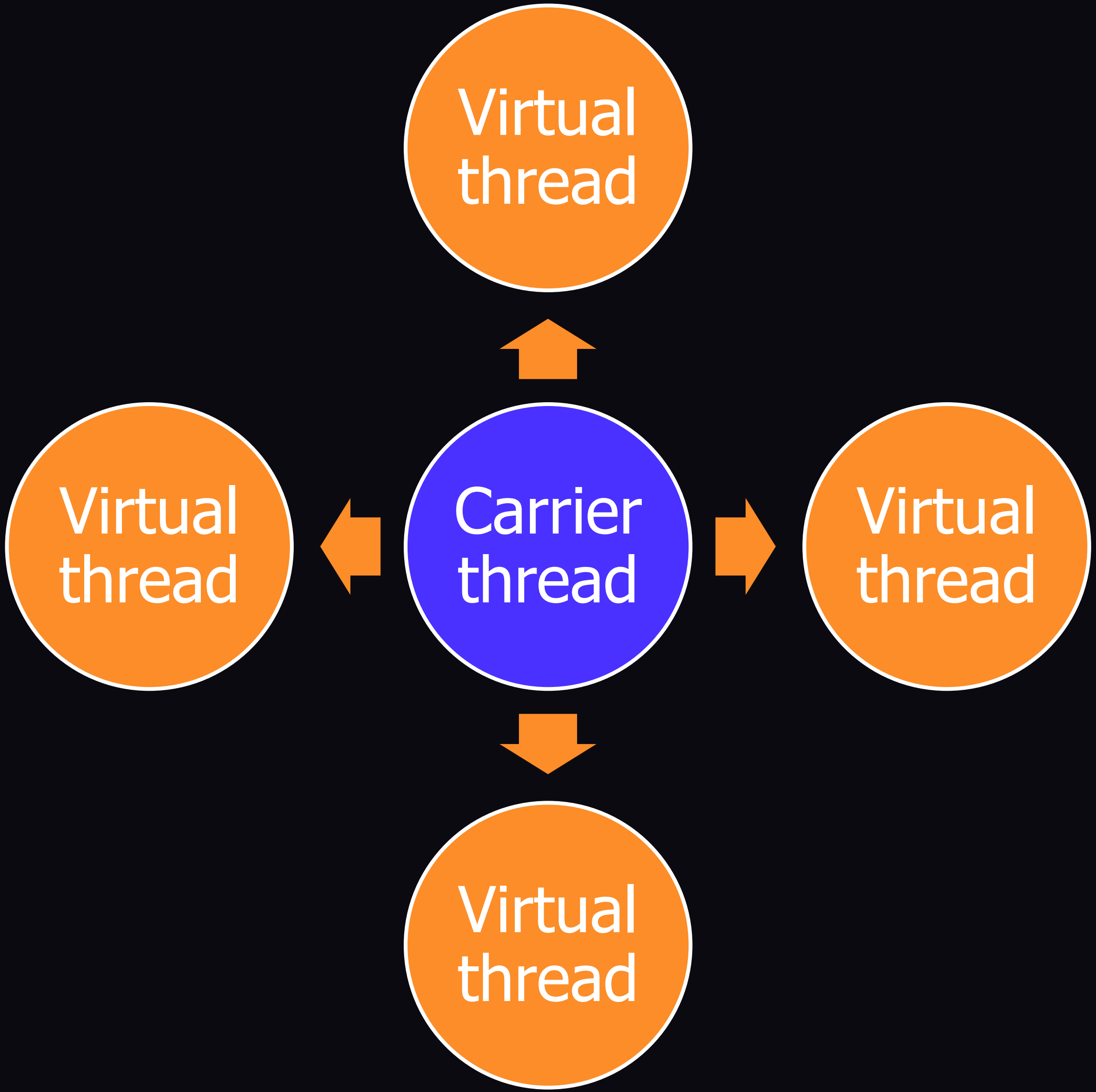
>>

MAX # os threads

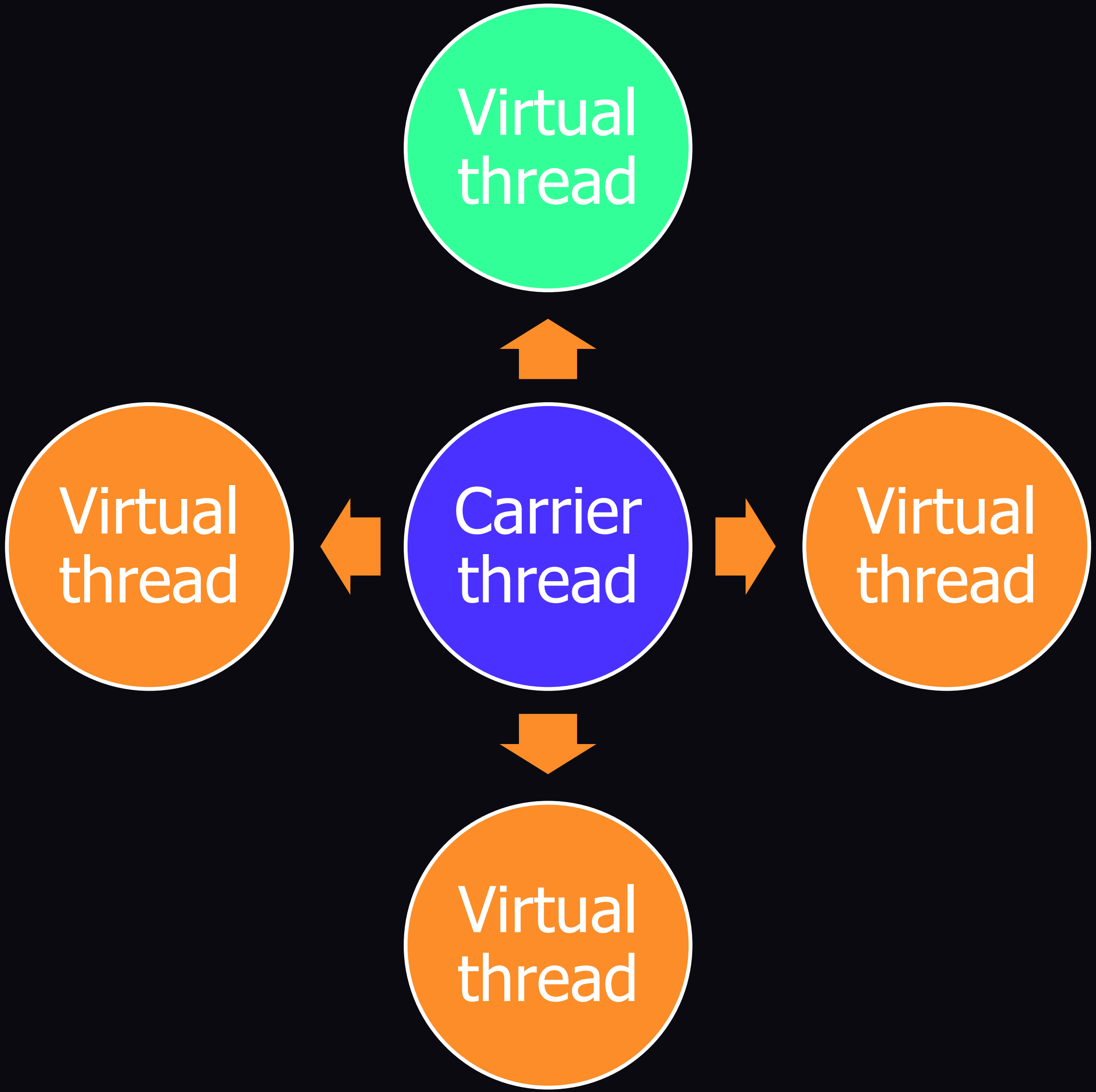
Virtual thread



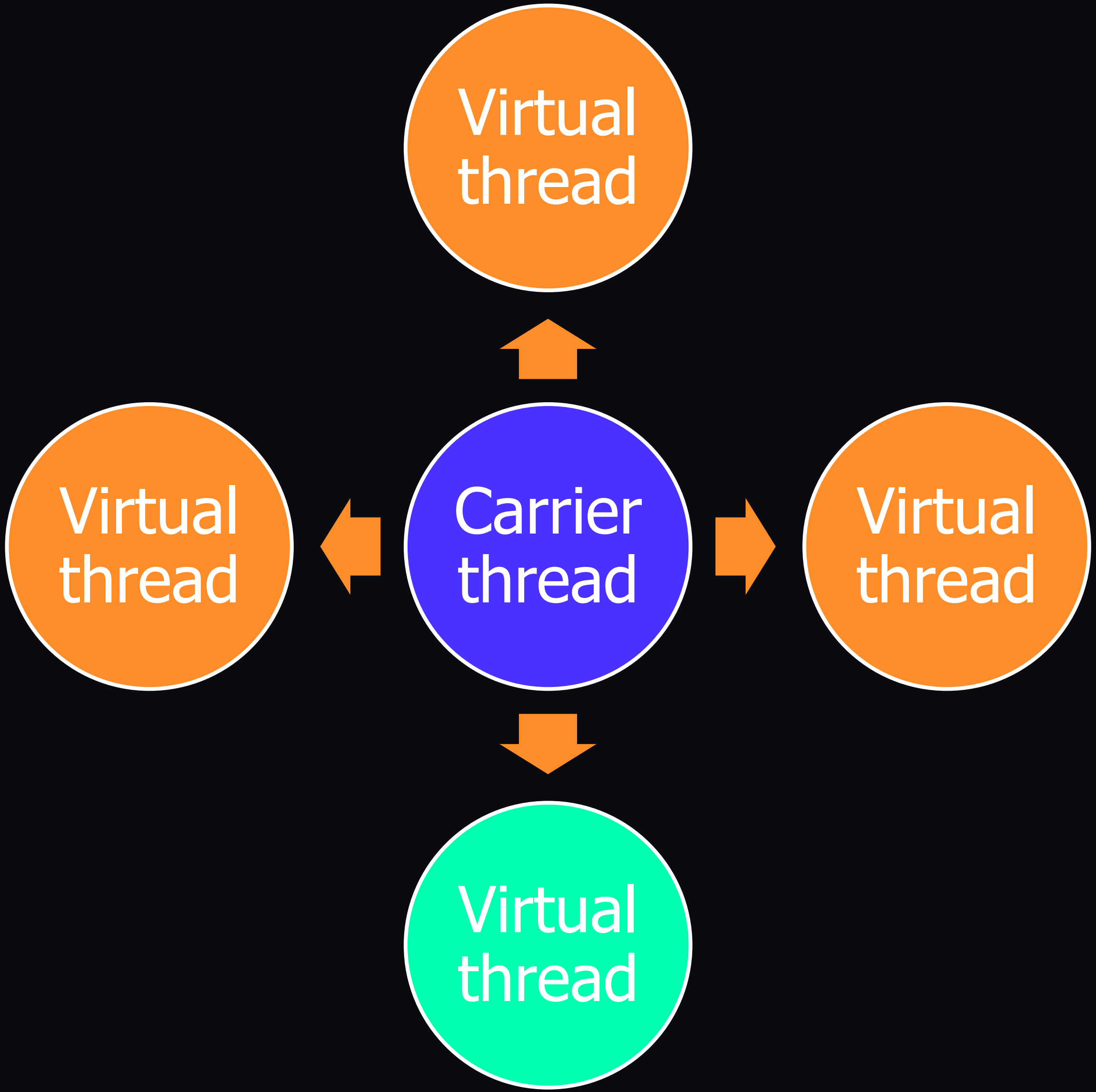
Virtual thread



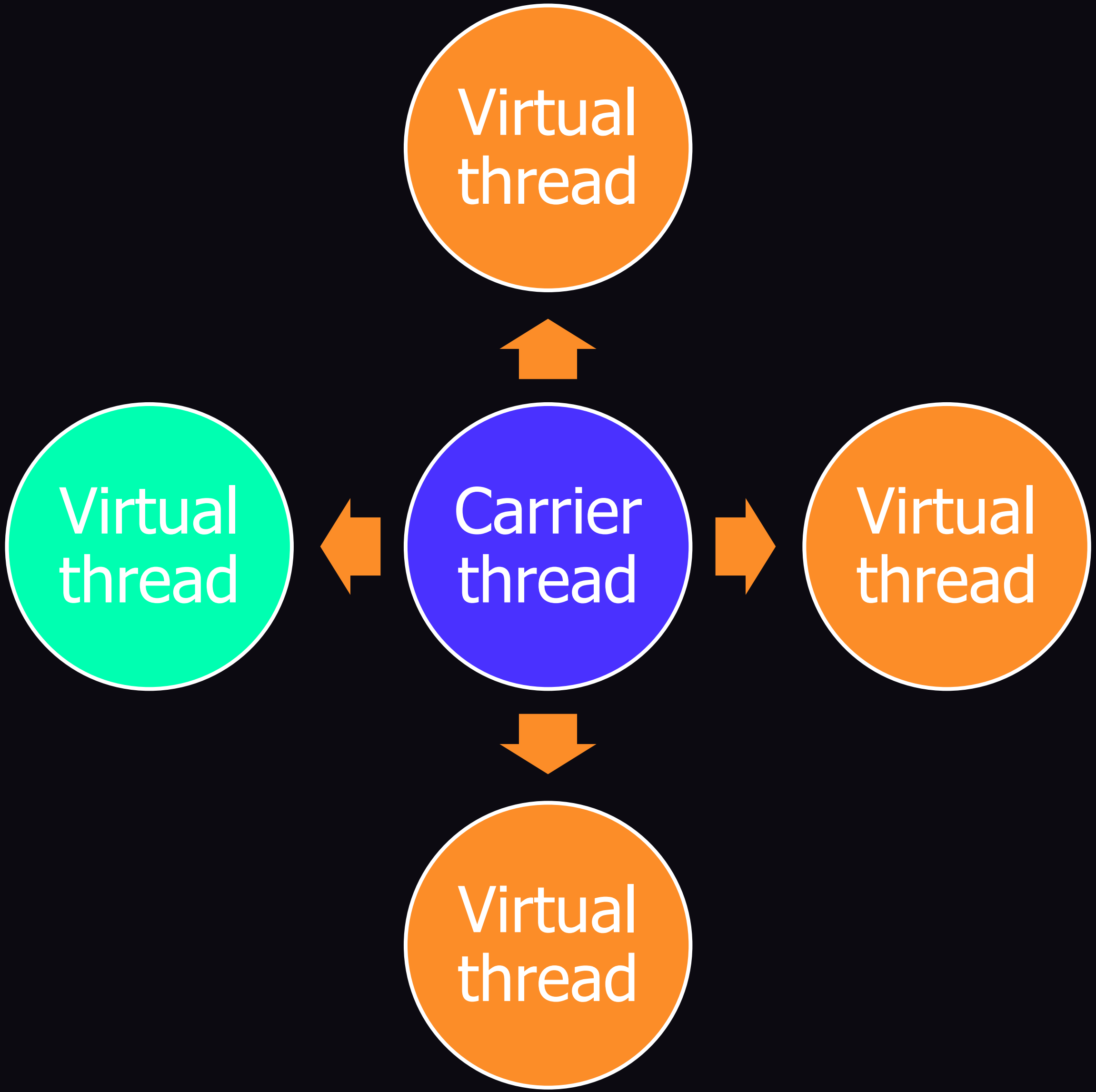
Virtual thread



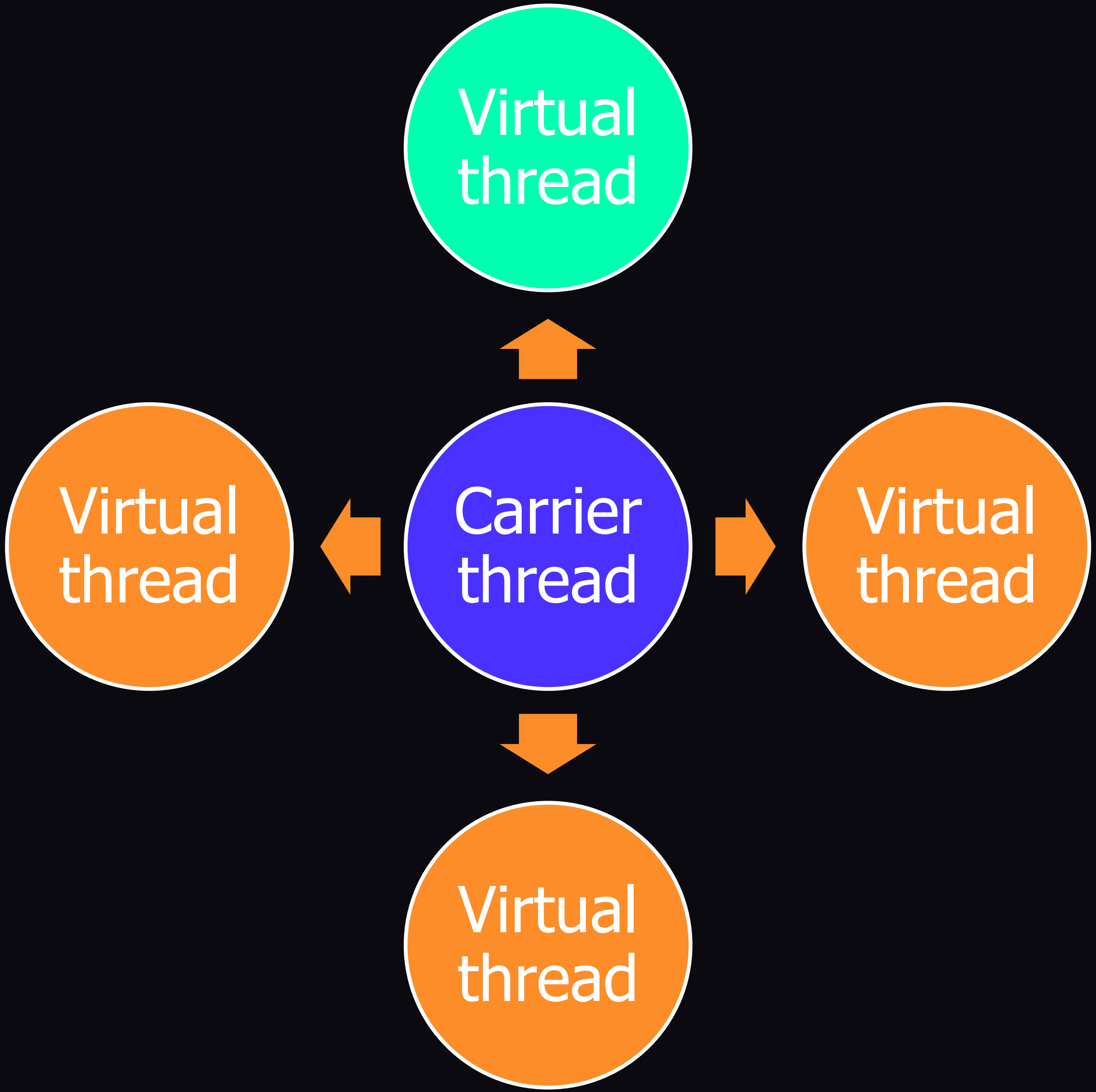
Virtual thread



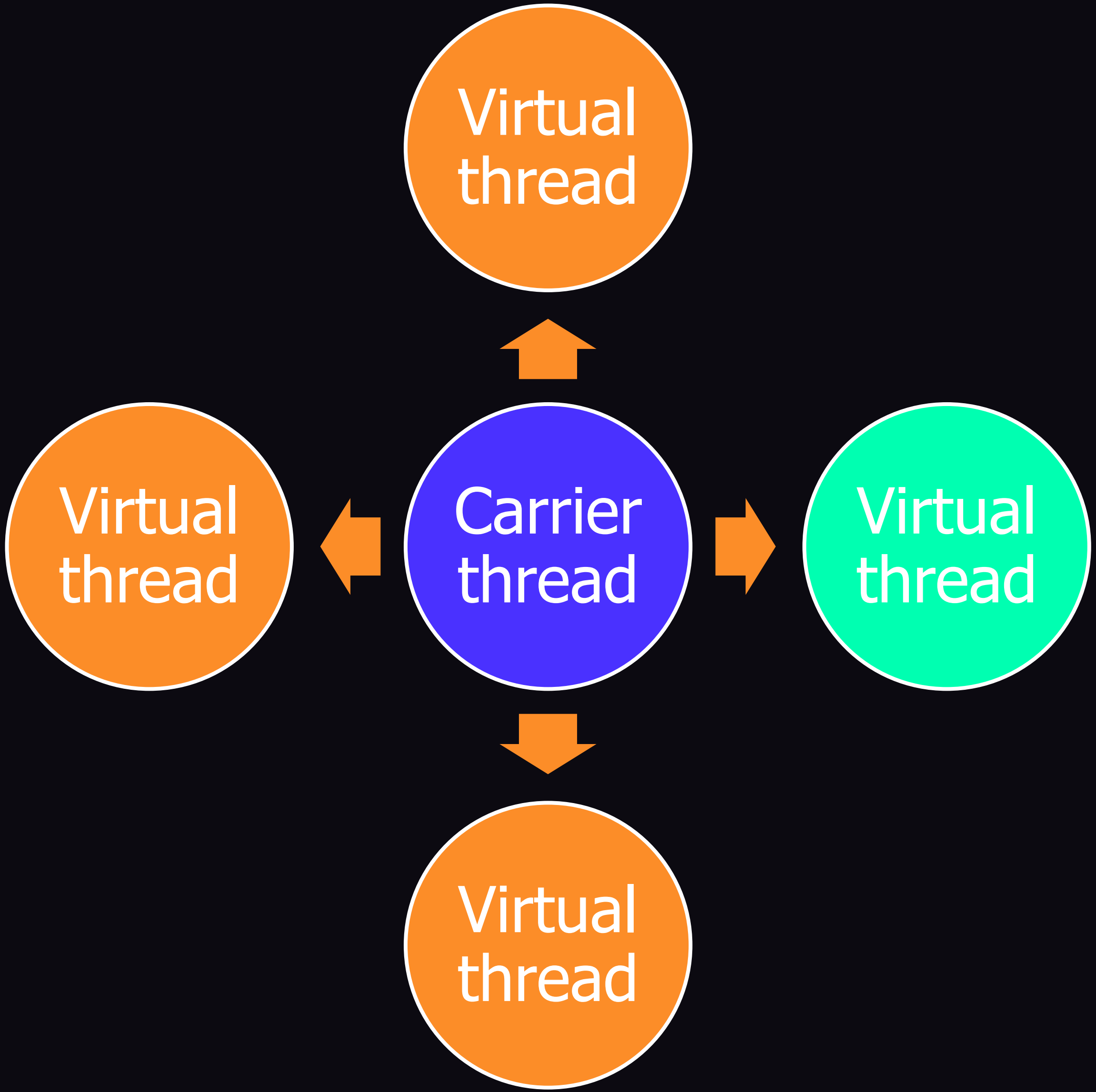
Virtual thread



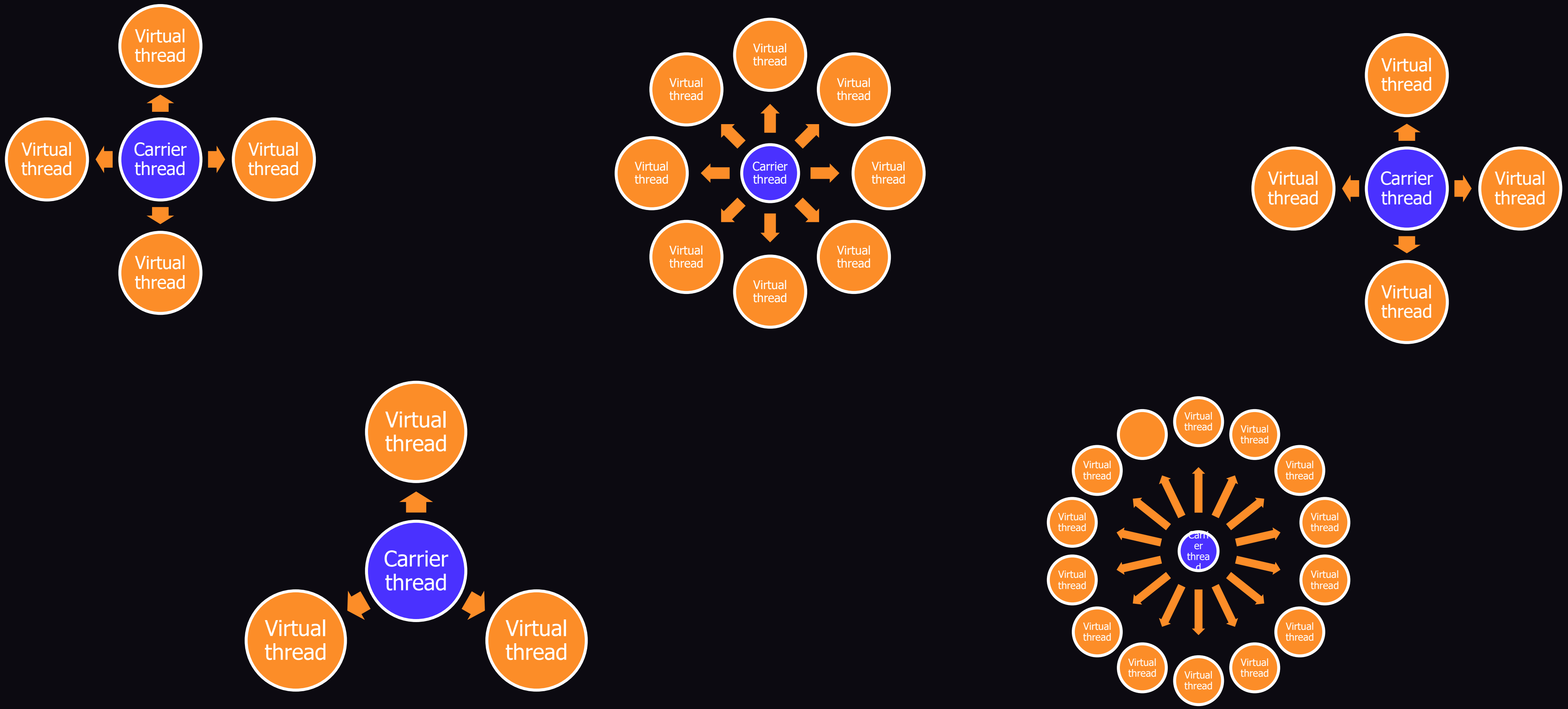
Virtual thread



Virtual thread



Virtual thread



Virtual thread

Which operations block thread?

JDK 19-21

Which operations block thread?

Classic (platform) thread

```
public static void main(String[] args) {
   IntStream.range(0, THREADS_NO)
         .forEach(i -> new Thread(new Task()).start());
}
```

Virtual thread

```
public static void main(String[] args) {
    IntStream.range(0, THREADS_NO)
            .forEach(i -> Thread.ofVirtual().start(new Task()));
}
```

Virtual thread

```
public static void main(String[] args) {
    IntStream.range(0, THREADS_NO)
            .forEach(i -> Thread.ofVirtual().start(new Task()));
}
```

Virtual thread

- ❗ Don’t use with thread pool
- ❗ It still can be blocked by using **synchronized**
- ❗ No priorities assignment

How is it useful in commercial projects?

We need a framework

Why framework?

- No need to reinvent a wheel
- Less code needed (less boilerplate code)
- Best practices in place
- Compatibility out of the box
- Gives everything (or almost everything ;)) that You need to create an application

STRUTS

MICRONAUT®

QUARKUS

JAKARTA® EE

GWT

spring

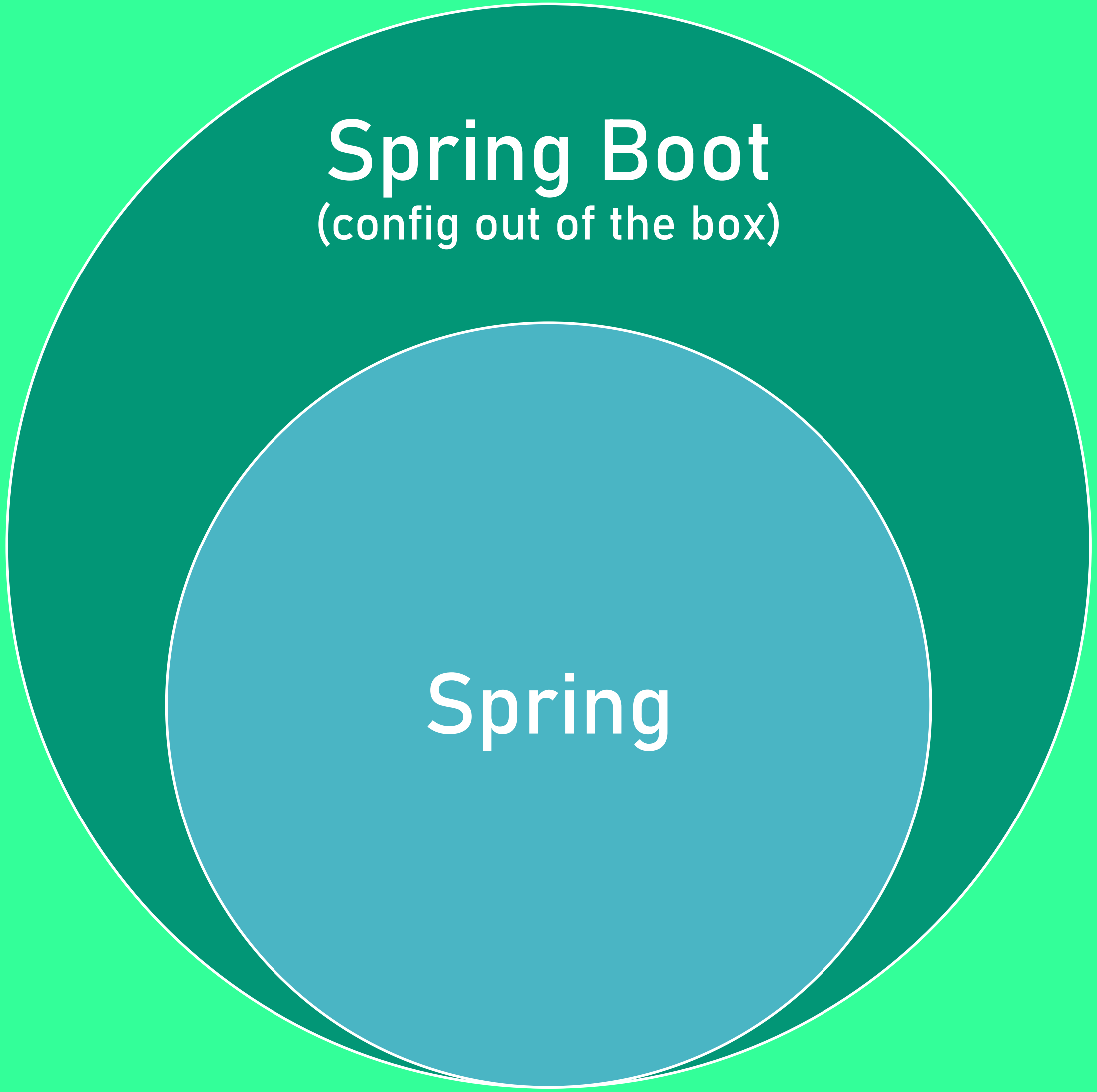
Spring framework

Why Spring?

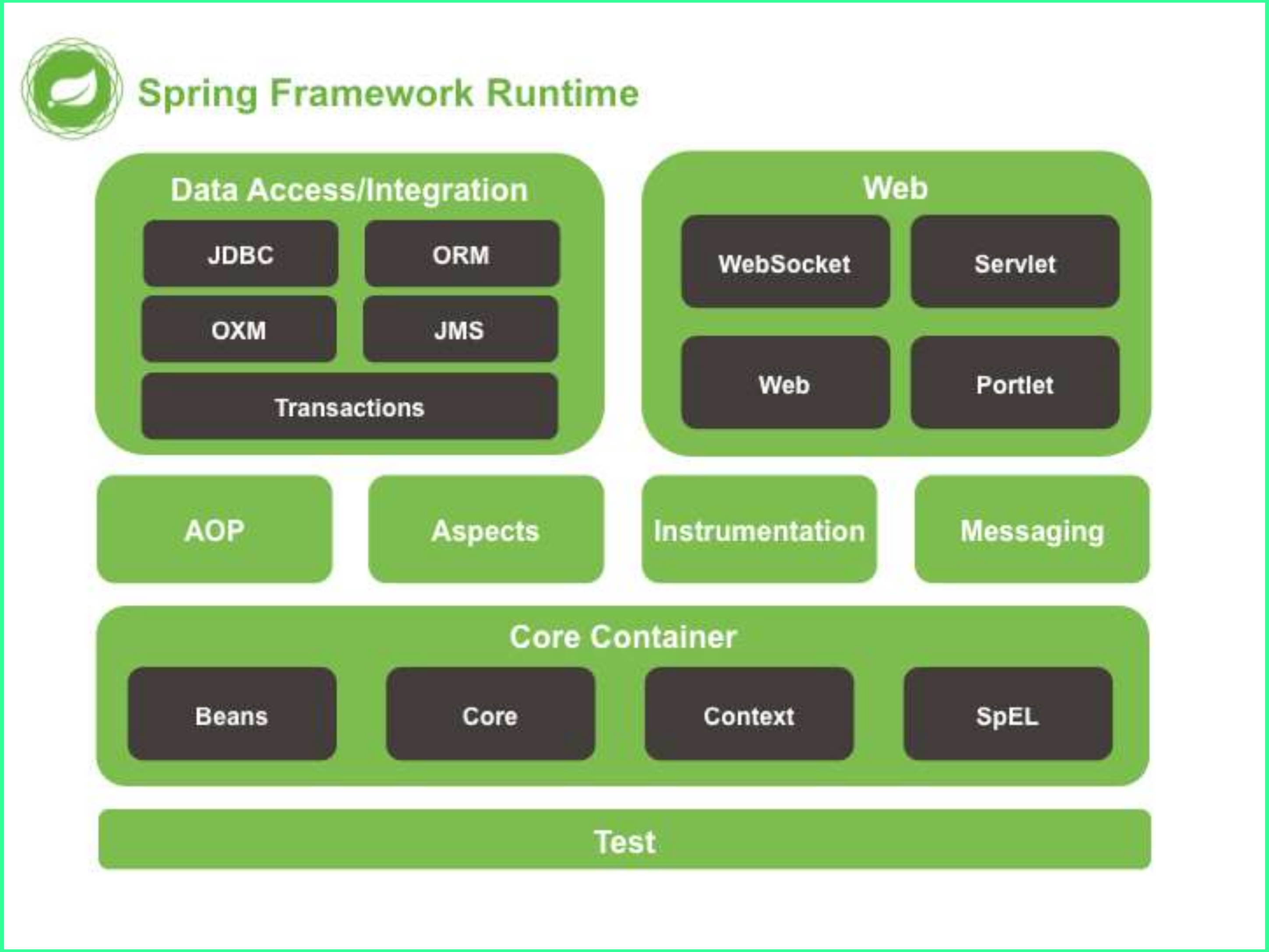
- Advanced framework
- Lot of modules/features
- Flexible
- Strong community
- Actively developed
- A lot of out of the box with Spring Boot

Spring vs Java EE (Jakarta EE)

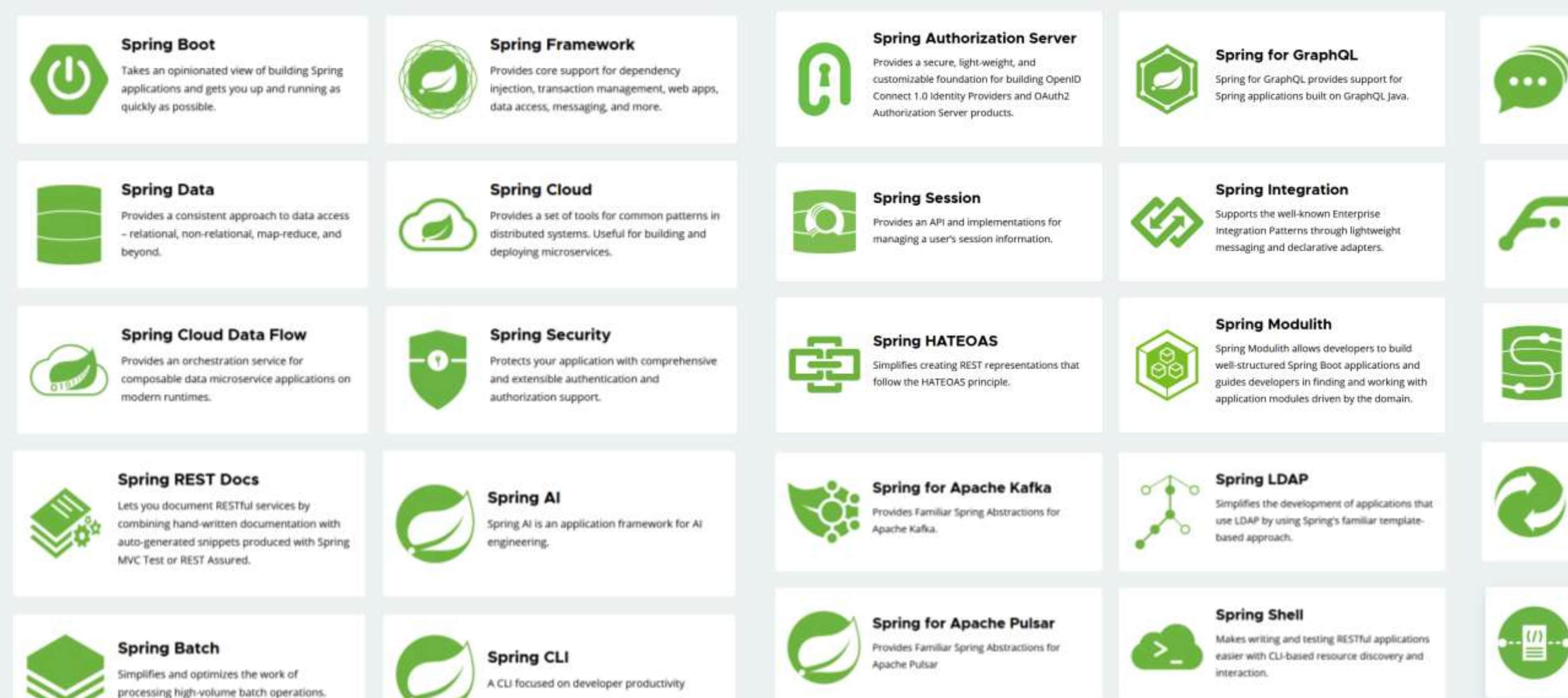
Spring vs Spring Boot



Spring modules



Spring projects



Spring Boot
Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.

Spring Framework
Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.

Spring Authorization Server
Provides a secure, light-weight, and customizable foundation for building OpenID Connect 1.0 Identity Providers and OAuth2 Authorization Server products.

Spring for GraphQL
Spring for GraphQL provides support for Spring applications built on GraphQL Java.

Spring Data
Provides a consistent approach to data access – relational, non-relational, map-reduce, and beyond.

Spring Cloud
Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.

Spring Session
Provides an API and implementations for managing a user's session information.

Spring Integration
Supports the well-known Enterprise Integration Patterns through lightweight messaging and declarative adapters.

Spring Cloud Data Flow
Provides an orchestration service for composable data microservice applications on modern runtimes.

Spring Security
Protects your application with comprehensive and extensible authentication and authorization support.

Spring HATEOAS
Simplifies creating REST representations that follow the HATEOAS principle.

Spring Modulith
Spring Modulith allows developers to build well-structured Spring Boot applications and guides developers in finding and working with application modules driven by the domain.

Spring REST Docs
Lets you document RESTful services by combining hand-written documentation with auto-generated snippets produced with Spring MVC Test or REST Assured.

Spring AI
Spring AI is an application framework for AI engineering.

Spring for Apache Kafka
Provides Familiar Spring Abstractions for Apache Kafka.

Spring LDAP
Simplifies the development of applications that use LDAP by using Spring's familiar template-based approach.

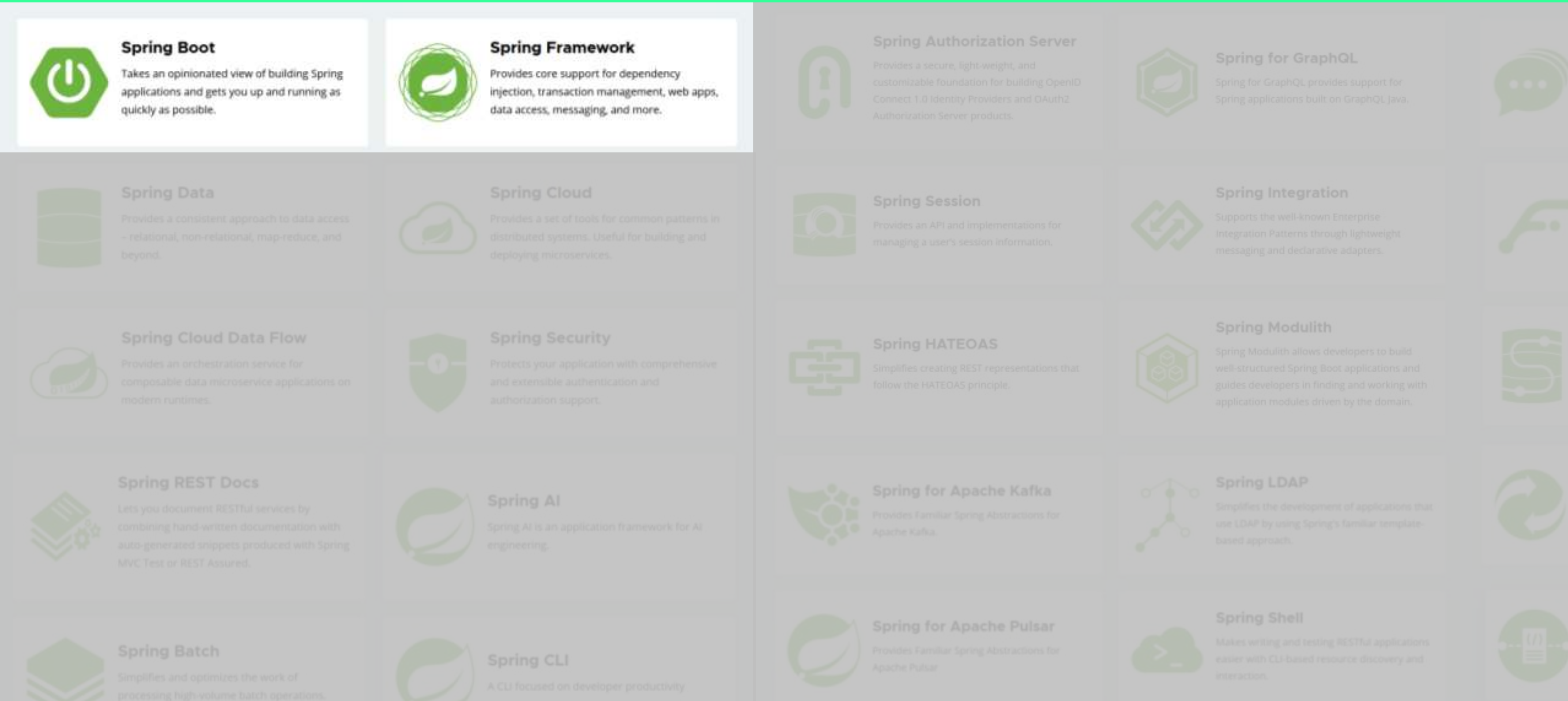
Spring Batch
Simplifies and optimizes the work of processing high-volume batch operations.

Spring CLI
A CLI focused on developer productivity

Spring for Apache Pulsar
Provides Familiar Spring Abstractions for Apache Pulsar

Spring Shell
Makes writing and testing RESTful applications easier with CLI-based resource discovery and interaction.

Spring projects



Spring Boot
Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.

Spring Framework
Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.

Spring Authorization Server
Provides a secure, light-weight, and customizable foundation for building OpenID Connect 1.0 Identity Providers and OAuth2 Authorization Server products.

Spring for GraphQL
Spring for GraphQL provides support for Spring applications built on GraphQL Java.

Spring Data
Provides a consistent approach to data access – relational, non-relational, map-reduce, and beyond.

Spring Cloud
Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.

Spring Session
Provides an API and implementations for managing a user's session information.

Spring Integration
Supports the well-known Enterprise Integration Patterns through lightweight messaging and declarative adapters.

Spring Cloud Data Flow
Provides an orchestration service for composable data microservice applications on modern runtimes.

Spring Security
Protects your application with comprehensive and extensible authentication and authorization support.

Spring HATEOAS
Simplifies creating REST representations that follow the HATEOAS principle.

Spring Modulith
Spring Modulith allows developers to build well-structured Spring Boot applications and guides developers in finding and working with application modules driven by the domain.

Spring REST Docs
Lets you document RESTful services by combining hand-written documentation with auto-generated snippets produced with Spring MVC Test or REST Assured.

Spring AI
Spring AI is an application framework for AI engineering.

Spring for Apache Kafka
Provides Familiar Spring Abstractions for Apache Kafka.

Spring LDAP
Simplifies the development of applications that use LDAP by using Spring's familiar template-based approach.

Spring Batch
Simplifies and optimizes the work of processing high-volume batch operations.

Spring CLI
A CLI focused on developer productivity

Spring for Apache Pulsar
Provides Familiar Spring Abstractions for Apache Pulsar

Spring Shell
Makes writing and testing RESTful applications easier with CLI-based resource discovery and interaction.

Basic annotations

- **@SpringBootApplication** – class level - basic Spring Boot annotation (aggregates a few other annotation that kickstarts spring application)
- **@RestController** – class level - creates rest controller (class with endpoint definitions)
- **@Get/Post/Put/Patch/DeleteMapping** – method level – creates http endpoint
- **@Repository** – interface level – creates data storage access point with useful default methods like (findById, findAll, save, etc.)

Spring Web

Spring Web

- For web based applications
- Allows creation of endpoints
- Classes that expose endpoints are called Controllers
- Supports REST standard

Sample rest controller

```
@RestController
public class StatusController {
    @GetMapping("/status/{versionNumber}")
    public String appStatus(@PathVariable int versionNumber) {
        if (versionNumber < 5) {
            return "Ok, deployed";
        }
        return "Not available";
    }
}
```

Sample rest controller

```
public class StatusController {







}
```

Sample rest controller

```
@RestController
public class StatusController {








}
```

Sample rest controller

```
@RestController
public class StatusController {

    public String appStatus(                    int versionNumber) {




    }
}
```

Sample rest controller

```
@RestController
public class StatusController {

    public String appStatus(                              int versionNumber) {
        if (versionNumber < 5) {
            return "Ok, deployed";
        }
        return "Not available";
    }
}
```

Sample rest controller

```
@RestController
public class StatusController {
  @GetMapping("/status/{versionNumber}")
  public String appStatus(                    int versionNumber) {
    if (versionNumber < 5) {
      return "Ok, deployed";
    }
    return "Not available";
  }
}
```

Sample rest controller

```
@RestController
public class StatusController {
   @GetMapping("/status/{versionNumber}")
   public String appStatus(@PathVariable int versionNumber) {
      if (versionNumber < 5) {
         return "Ok, deployed";
      }
      return "Not available";
   }
}
```

Sample rest controller – responses

```
curl -X GET localhost:8080/status/1
Ok, deployed
curl -X GET localhost:8080/status/2
Ok, deployed
curl -X GET localhost:8080/status/5
Not available
```

Data base

Spring Data

- Available dependencies:
    - JDBC (plain data base access)
    - JPA (Java Persistence API)
    - Elasticsearch
    - Reactive relational db connections
    - Reactive Redis
    - MongoDB

Spring Data JPA

- Straight forward, boilerplate-less solution for db connections
- By default uses Hibernate ORM
- Out of the box db access methods for CRUD operations

Spring Data JPA - entity

Customer.java

```
public class Customer {



  private Long id;
  private String name;

  // getters and setters
```

Spring Data JPA – entity

Customer.java

```
@Entity
public class Customer {



    private Long id;
    private String name;

    // getters and setters
```

Spring Data JPA – entity

Customer.java

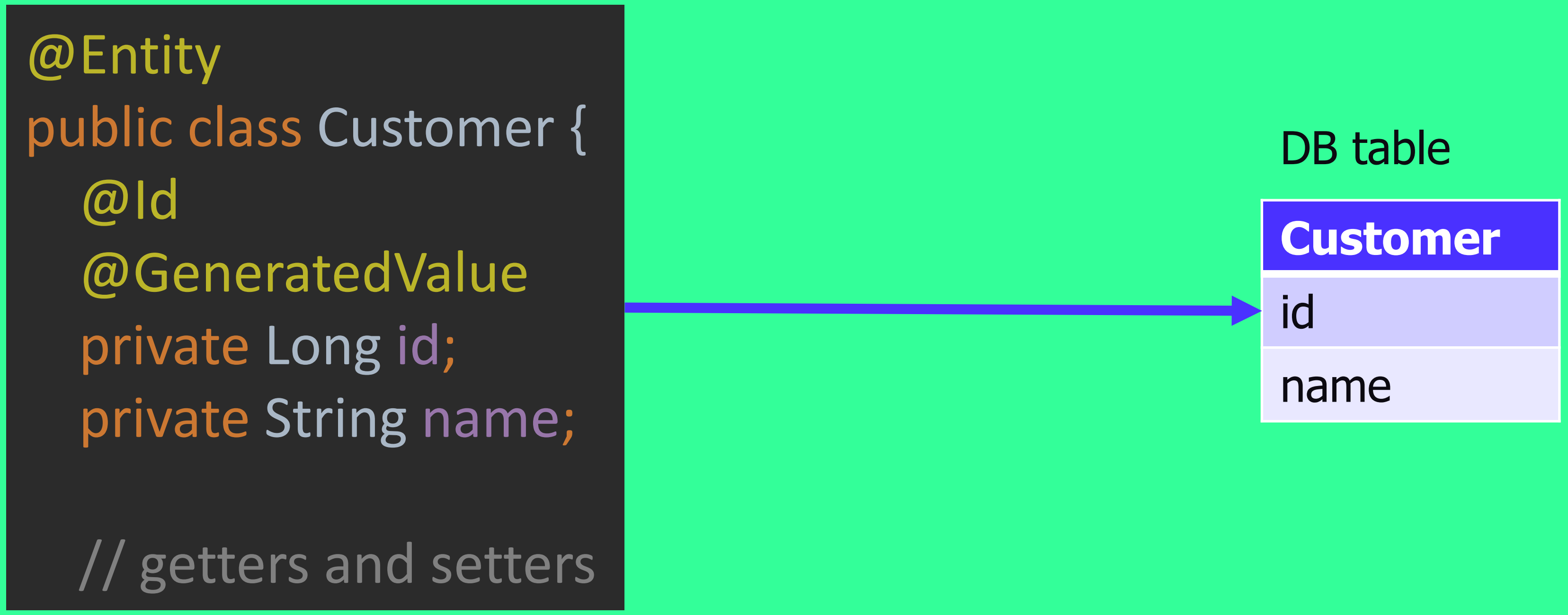
```java
@Entity
public class Customer {
   @Id

   private Long id;
   private String name;

   // getters and setters
```

Spring Data JPA – entity

Customer.java

```java
@Entity
public class Customer {
   @Id
   @GeneratedValue
   private Long id;
   private String name;

   // getters and setters
```

Spring Data JPA - entity



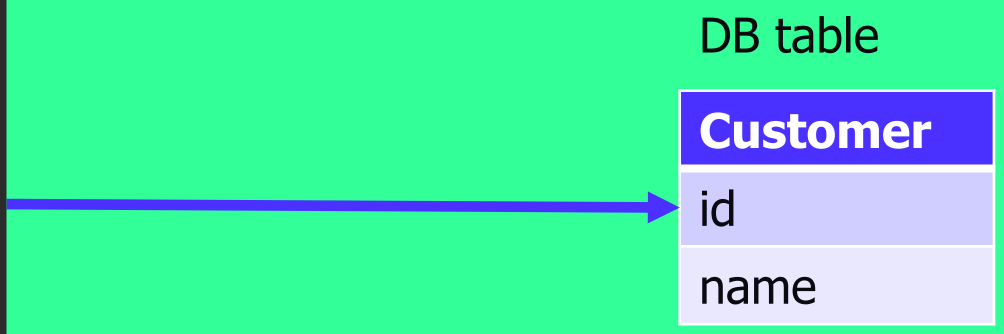
Customer.java

```java
@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;
    private String name;

    // getters and setters
```

Spring Data JPA - entity

Customer.java

```java
@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;
    private String name;

    // getters and setters
```



CustomerRepository.java

```java
@Repository
public interface CustomerRepository extends CrudRepository< Customer, Long> {}
```

Spring Data JPA – entity

Customer.java

```
@Entity
public class Customer {
  @Id
  @GeneratedValue
  private Long id;
  private String name;

  // getters and setters
```

Makes class visible by JPA

Field below @Id annotation is treated as db table id

JPA will automatically give random, unique value for field below

CustomerRepository.java

```
@Repository
public interface CustomerRepository extends CrudRepository< Customer, Long> {}
```

```java
public interface CrudRepository<T, ID> extends Repository<T, ID> {

   <S extends T> S save(S entity);
   <S extends T> Iterable<S> saveAll(Iterable<S> entities);

   Optional<T> findById(ID id);
   boolean existsById(ID id);
   Iterable<T> findAll();
   Iterable<T> findAllById(Iterable<ID> ids);

   long count();

   void deleteById(ID id);
   void delete(T entity);
   void deleteAllById(Iterable<? extends ID> ids);
   void deleteAll(Iterable<? extends T> entities);
   void deleteAll();
}
```

CrudRepository behind the scenes

What about the db?

Adding config to **application.properties** is enough

Sample db config for MySql db

```
spring.datasource.url=jdbc:mysql://localhost:3306/db_example

spring.datasource.username=springuser

spring.datasource.password=ThePassword

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update
```

Sample db config for MySql db

```
spring.datasource.url=jdbc:mysql://localhost:3306/db_example
```

Sample db config for MySql db

```
spring.datasource.url=jdbc:mysql://localhost:3306/db_example

spring.datasource.username=springuser

spring.datasource.password=ThePassword
```

Sample db config for MySql db

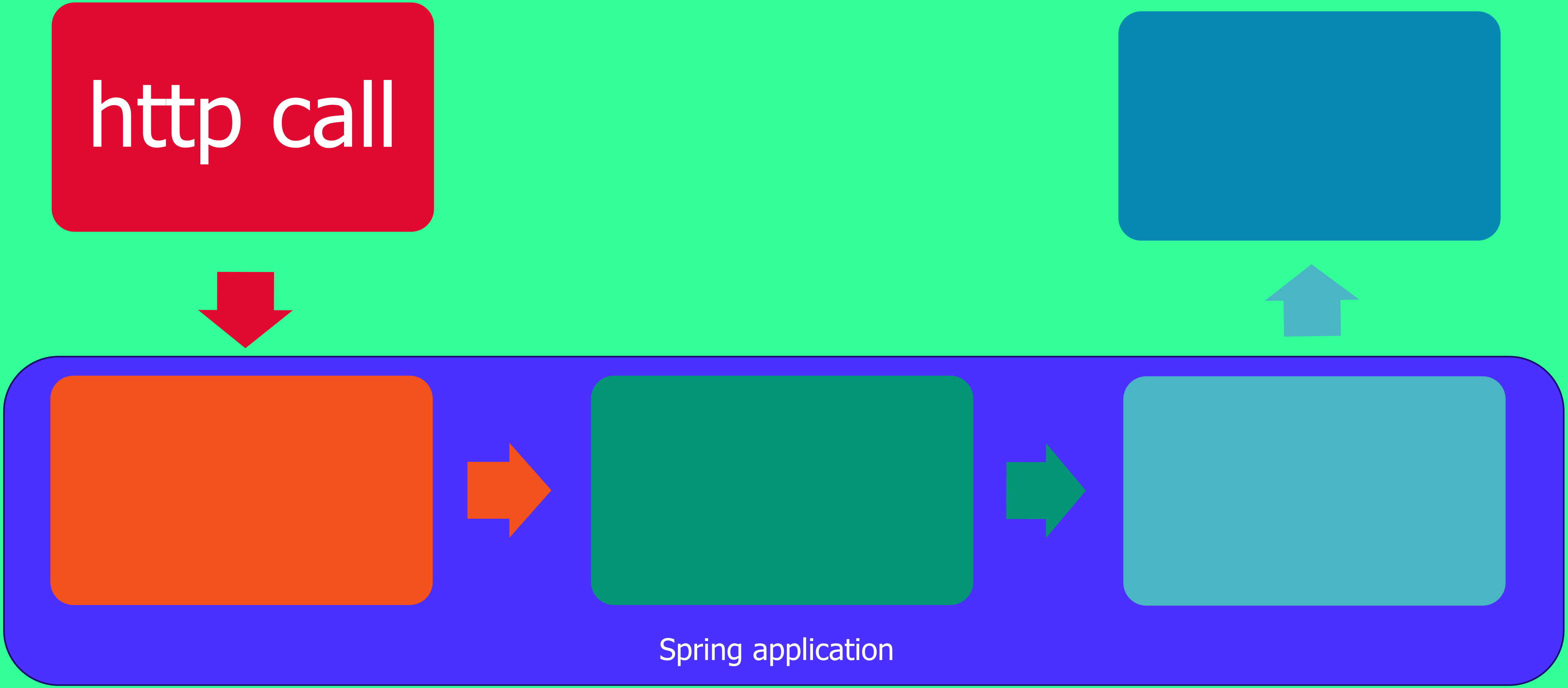
```
spring.datasource.url=jdbc:mysql://localhost:3306/db_example

spring.datasource.username=springuser

spring.datasource.password=ThePassword

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

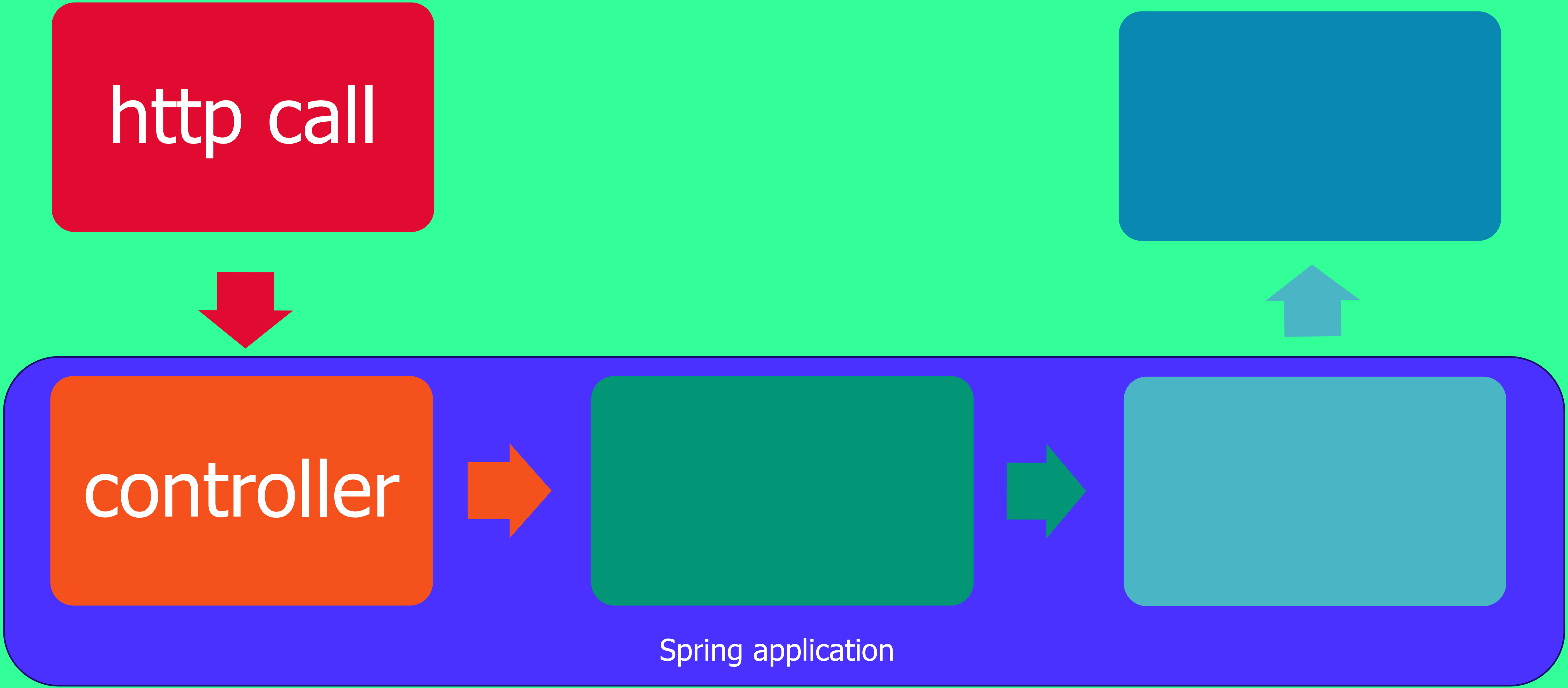
Sample db config for MySql db

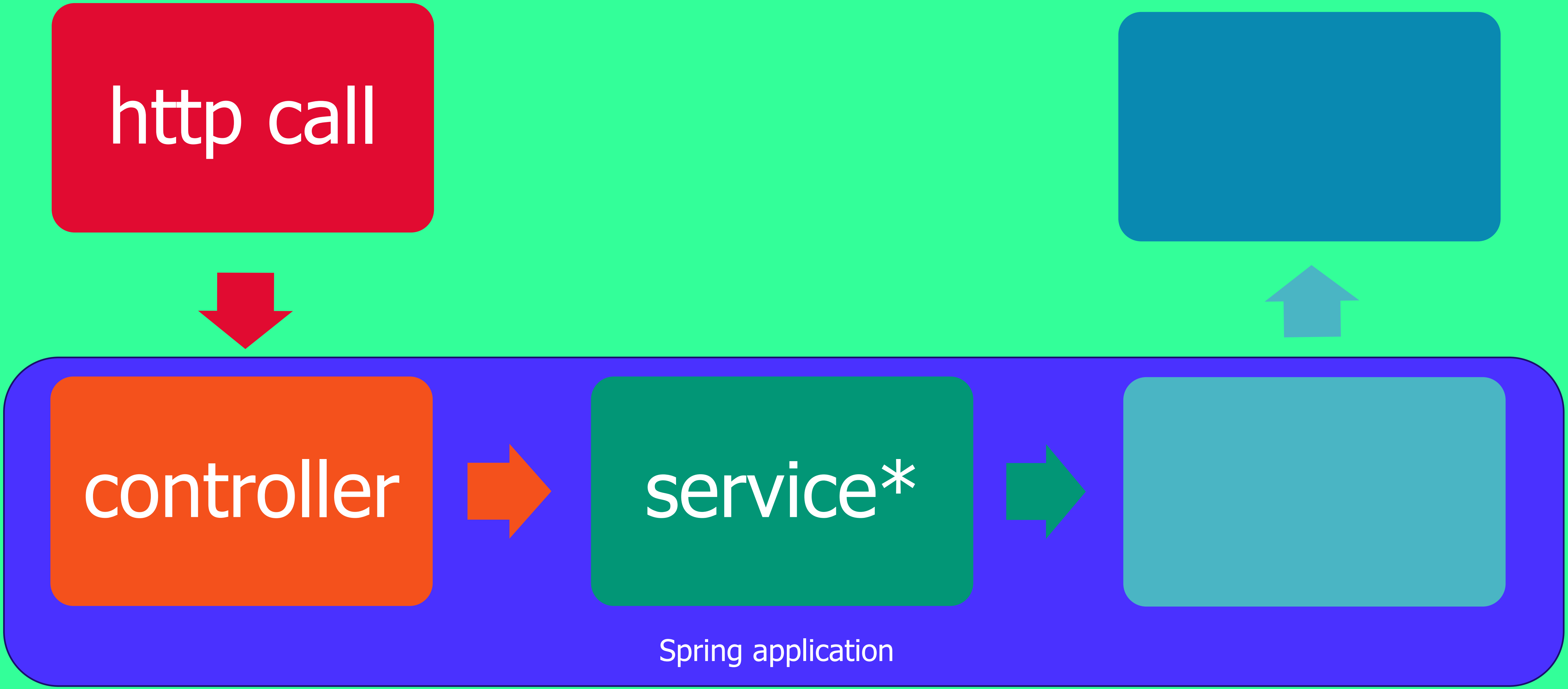
```
spring.datasource.url=jdbc:mysql://localhost:3306/db_example

spring.datasource.username=springuser

spring.datasource.password=ThePassword

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update
```

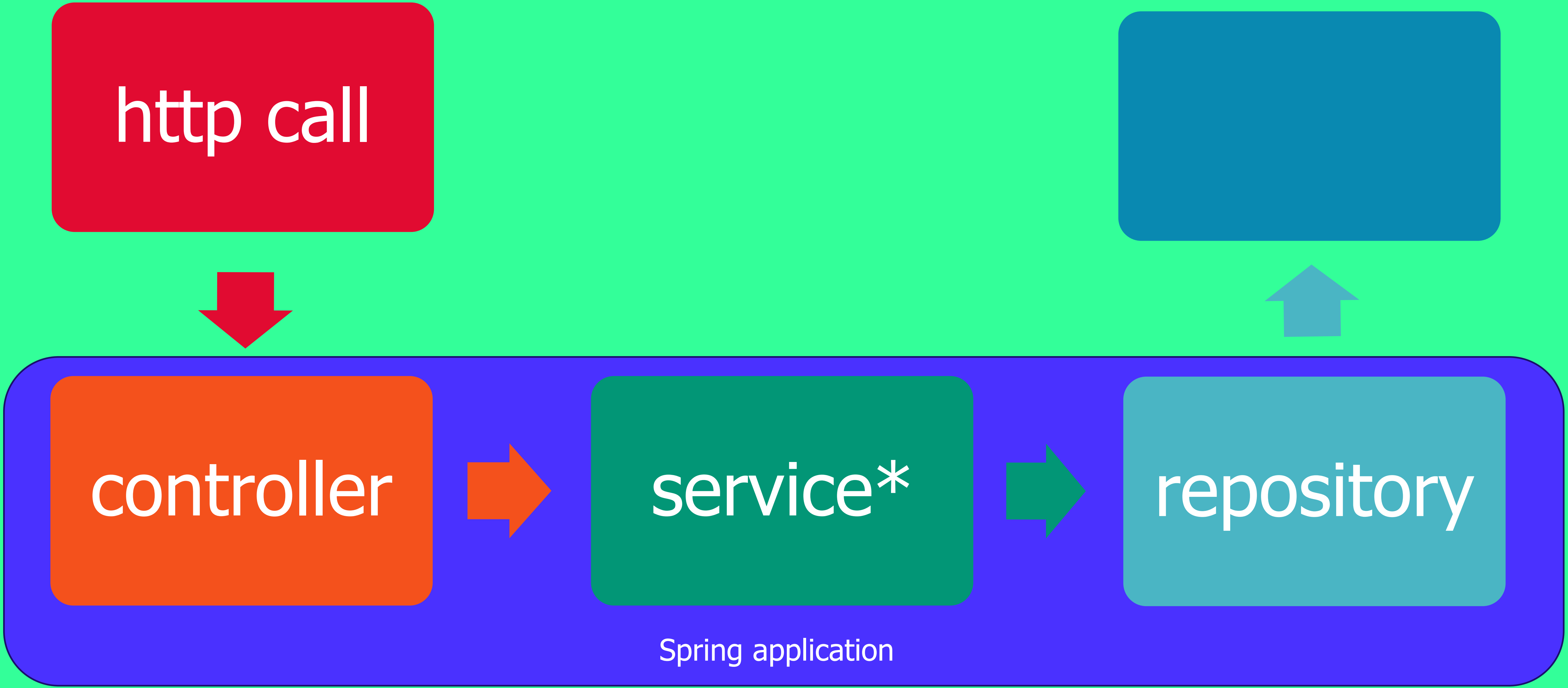
H2 – in memory database

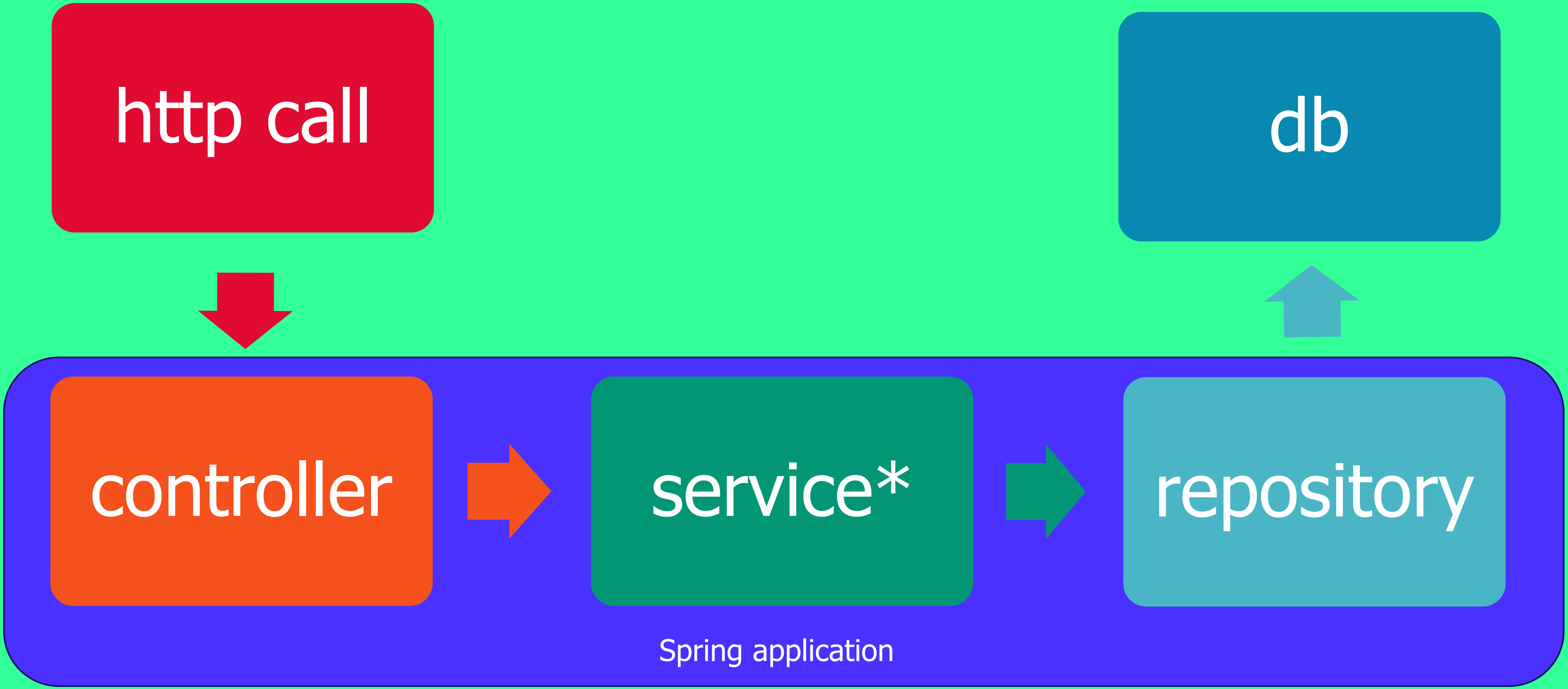
- Simple
- Fast
- After application is closed, all data vanishes
- Good for prototyping
- Good for tests
- No configuration needed, when used with Spring Boot and Spring Data Jpa


http call
Spring application


http call
controller
Spring application


http call
controller
service*
Spring application


http call
controller
service*
repository
Spring application


http call
db
controller
service*
repository
Spring application

What’s common for (Rest)Controller, Service, Repository?

They are Beans

Visible by Spring IoC (Inversion of Control) container

Java – creation of objects

```java
String str = "string";
```

```java
int a = 5;
float num = 5.2;
```

```java
Object o = new Object();
MyClass klass = new MyClass();
```

bean

In Spring ~~object~~ creation is automatic

How to turn an object into a bean?

- Use annotation (like @Component, @Controller, @Service, @Repository, etc.) above class definition

- Use configuration:
  - In Java (Spring configuration class – with @Configuration annotation)
  - In XML (xml file) – this is rather legacy solution

How do beans communicate?

Just a plain java composition

```
@Service
public class CustomerService {

   private CustomerRepository customerRepository;

   public void saveCustomer(Customer customer) {
      customerRepository.save(customer);
   }
//... omitted
```

How to inject dependency?

- Constructor
- Setter
- Field (avoid)

Dependency injection by constructor

```
@Service
public class CustomerService {

    private CustomerRepository customerRepository;

    public CustomerService(CustomerRepository customerRepository) {
        this.customerRepository = customerRepository;
    }
```

Dependency injection by setter

```
@Service
public class CustomerService {

    private CustomerRepository customerRepository;

    @Autowired
    public void setCustomerRepository(CustomerRepository customerRepository) {
        this.customerRepository = customerRepository;
    }
```

Dependency injection by field

```
@Service
public class CustomerService {

    @Autowired
    private CustomerRepository customerRepository;
```

AVOID*

*Difficult to unit test

Where to keep application properties?

application.properties file

Application.properties

$$\overbrace{\texttt{spring.application.name}}^{\text{Key}}\texttt{=}\underbrace{\texttt{demo}}_{\text{Value}}$$

How to use properties in Java code?

application.properties

```
spring.application.name=demo
```

HelloController.java

```
@RestController
public class HelloController {

    @Value("${spring.application.name}")
    private String appName;
```

How to start with Spring?

Spring Initializr at:

start.spring.io

mbaranowski.pl