

A large, vibrant red brushstroke graphic that starts on the left side of the slide and sweeps across to the right, curving upwards at the end. The stroke has a textured, hand-painted appearance with visible bristles and varying thicknesses.

Java 17
Your Code With Even More Quality
HELBER BELMIRO

\$ ~ whoami
Helber Belmiro

I help Java engineers create APIs that are easy to evolve and upgrade so they can write maintainable software and stay up-to-date with technology.

Software Engineer at  **Red Hat**

Working on:



Kogito



jBPM

Blog: thegreatapi.com

Twitter: [@helber_belmiro](https://twitter.com/helber_belmiro)

LinkedIn: linkedin.com/in/hbelmiro

Where do Bugs Come From?

Among others...

- Non-intuitive APIs
- Code that is hard to read
- Verbosity
- Mutable code and objects

Newer versions of Java

Java 17

- More intuitive APIs
- More readable
- Less verbose code
- Immutable code and objects

Newer versions of Java

Java 17

- More intuitive APIs
- More readable
- Less verbose code
- Immutable code and objects

Less bugs!

Versions of Java - History

Java 8 (1.8) – March 2014 – LTS

Java 9 – September 2017

Java 10 – March 2018

Java 11 – September 2018 – LTS

Java 12 – March 2019

Java 13 – September 2019

Java 14 – March 2020

Java 15 – September 2020

Java 16 – March 2021

Java 17 – September 2021 – LTS

✨ Java 18 – March 2022 ✨

Versions of Java - History

Java 8 (1.8) – March 2014 – LTS

Java 9 – September 2017

Java 10 – March 2018

Java 11 – September 2018 – LTS

Java 12 – March 2019

Java 13 – September 2019

Java 14 – March 2020

Java 15 – September 2020

Java 16 – March 2021

Java 17 – September 2021 – LTS

✨ Java 18 – March 2022 ✨

What version
are you using?

16645 Bugs Fixed

Java Bug System

Project: JDK

Type: Bug


Fix Version: 10, 11, 12, 13, 14, 15, 16, 17, 9

Status: Resolved

Resolution: Fixed

Sorted by: Key descending

1-1000 of **16645** as at: **2022-03-16 03:45**

T	Key	Summary
	JDK-8273592	Backout JDK-8271868

<https://bugs.openjdk.java.net/sr/jira.issueviews:searchrequest-printable/temp/SearchRequest.html?jqlQuery=project+%3D+JDK+AND+issuetype+%3D+Bug+AND+status+%3D+Resolved+AND+resolution+%3D+Fixed+AND+fixVersion+in+%28%2217%22%2C+%2210%22%2C+%2211%22%2C+%2212%22%2C+%2213%22%2C+%2214%22%2C+%2215%22%2C+%2216%22%2C+%229%22%29&tempMax=1000>

Java 8 – Creating a List

- How can I create a List with fixed elements?
- Can I pass the elements in constructor?

Java 8 – Creating a List - ArrayList Constructor

3 Overloads

@Range int initialCapacity

<no parameters>

@NotNull @Nonnull Collection<? extends String> c

```
final List<String> list = new ArrayList<>( $\emptyset$ );
```

Java 8 – Creating a List

```
final List<String> list = new ArrayList<>();  
list.add("This");  
list.add("is");  
list.add("a");  
list.add("list");
```

Java 8 – Creating a List

```
final List<String> list = new ArrayList<>();  
list.add("This");  
list.add("is");  
list.add("a");  
list.add("list");
```

Modifiable list!

Java 8 – Making the List Unmodifiable

```
final List<String> list = new ArrayList<>();  
list.add("This");  
list.add("is");  
list.add("a");  
list.add("list");  
list = Collections.unmodifiableList(list);
```

Java 8 – Making the List Unmodifiable

```
final List<String> list = new ArrayList<>();  
list.add("This");  
list.add("is");  
list.add("a");  
list.add("list");  
list = Collections.unmodifiableList(list);
```

Cannot assign a value to final variable 'list' ⋮

Make 'list' not final ↵ ↶ ↷ More actions... ⌘1

```
final List<String> list = new ArrayList<String>() ⋮
```

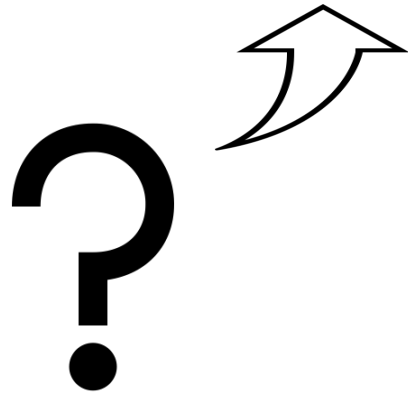
Immutable object, but mutable code

Java 8 – Creating a List - Alternative

```
final List<String> list = Arrays.asList("This", "is", "a", "list");
```

Java 8 – Creating a List - Alternative

```
final List<String> list = Arrays.asList("This", "is", "a", "list");
```



Java 9 – Creating a List

```
final List<String> list = List.of("This", "is", "a", "list");
```

Java 9 – Creating a List/Set/Map

```
final List<String> list = List.of("This", "is", "a", "list");  
  
final Set<String> set = Set.of("This", "is", "a", "set");  
  
final Map<Integer, String> map = Map.of(  
    1, "one",  
    2, "two",  
    3, "three"  
);
```

Java 9 – Modules

- APIs are like stars (20 API Paradoxes - Jaroslav Tulach)
- To keep evolving an API
- Expose only the API
- Encapsulate the implementations

Java 9 – Modules

Allows us to specify what packages we want to expose

- **Name** – the name of our module
- **Dependencies** – a list of other modules that this module depends on
- **Public Packages** – a list of all packages we want accessible from outside the module
- **Services Offered** – we can provide service implementations that can be consumed by other modules
- **Services Consumed** – allows the current module to be a consumer of a service
- **Reflection Permissions** – explicitly allows other classes to use reflection to access the private members of a package

<https://www.baeldung.com/java-9-modularity>

Java 10 – Copying Collections

Java 8

```
List<String> list = Collections.unmodifiableList(new ArrayList<>(originalList));  
Set<Object> set = Collections.unmodifiableSet(new HashSet<>(originalSet));  
Map<Object, Object> map = Collections.unmodifiableMap(new HashMap<>(originalMap));
```

Java 10 – Copying Collections

Java 8

```
List<String> list = Collections.unmodifiableList(new ArrayList<>(originalList));  
Set<Object> set = Collections.unmodifiableSet(new HashSet<>(originalSet));  
Map<Object, Object> map = Collections.unmodifiableMap(new HashMap<>(originalMap));
```

Java 10

```
List<String> list = List.copyOf(originalList);  
Set<Object> set = Set.copyOf(originalSet);  
Map<Object, Object> map = Map.copyOf(originalMap);
```

Java 10 – Collecting an Unmodifiable List/Set/Map from a Stream

Java 8

```
List<String> list = Collections.unmodifiableList(streamForList.collect(Collectors.toList()));
```

```
Set<String> set = Collections.unmodifiableSet(streamForSet.collect(Collectors.toSet()));
```

```
Map<String, String> map = Collections.unmodifiableMap(pointStream.collect(Collectors.toMap(  
    point -> point.getX().toString(),  
    point -> point.getY().toString()  
)));
```

Java 10 – Collecting an Unmodifiable List/Set/Map from a Stream

Java 8

```
List<String> list = Collections.unmodifiableList(streamForList.collect(Collectors.toList()));
```

```
Set<String> set = Collections.unmodifiableSet(streamForSet.collect(Collectors.toSet()));
```

```
Map<String, String> map = Collections.unmodifiableMap(pointStream.collect(Collectors.toMap(  
    point -> point.getX().toString(),  
    point -> point.getY().toString()  
)));
```

Java 10

```
List<String> list = streamForList.collect(Collectors.toUnmodifiableList());
```

```
Set<String> set = streamForSet.collect(Collectors.toUnmodifiableSet());
```

```
Map<String, String> map = pointStream.collect(Collectors.toUnmodifiableMap(  
    point -> point.getX().toString(),  
    point -> point.getY().toString()  
));
```


Java 11 – Repeating Strings

Java 8

```
public String repeat(String s) {  
    StringBuilder stringBuilder = new StringBuilder();  
    for (int i = 0; i < 4; i++) {  
        stringBuilder.append(s);  
    }  
    return stringBuilder.toString();  
}
```

Java 11 – Repeating Strings

Java 8

```
public String repeat(String s) {  
    StringBuilder stringBuilder = new StringBuilder();  
    for (int i = 0; i < 4; i++) {  
        stringBuilder.append(s);  
    }  
    return stringBuilder.toString();  
}
```

Java 11

```
public String repeat(String s) {  
    return s.repeat(4);  
}
```

Java 12 – Indenting (Multi-line) Strings

Java 8

```
public String indentation(String s, int indentationSize) {  
    return Stream.of(s.split(System.lineSeparator()))  
        .map(line -> {  
            StringBuilder stringBuilder = new StringBuilder();  
            for (int i = 0; i < indentationSize; i++) {  
                stringBuilder.append(' ');  
            }  
            stringBuilder.append(line);  
            return stringBuilder.toString();  
        }).collect(  
            Collectors.joining(  
                System.lineSeparator(),  
                prefix: "",  
                System.lineSeparator()  
            )  
        );  
}
```

Java 12 – Indenting (Multi-line) Strings

Java 8

```
public String indentation(String s, int indentationSize) {  
    return Stream.of(s.split(System.lineSeparator()))  
        .map(line -> {  
            StringBuilder stringBuilder = new StringBuilder();  
            for (int i = 0; i < indentationSize; i++) {  
                stringBuilder.append(' ');  
            }  
            stringBuilder.append(line);  
            return stringBuilder.toString();  
        }).collect(  
            Collectors.joining(  
                System.lineSeparator(),  
                prefix: "",  
                System.lineSeparator()  
            )  
        );  
}
```

Java 11

```
public String indentation(String s, int indentationSize) {  
    return s.lines()  
        .map(line -> " ".repeat(indentationSize) + line)  
        .collect(  
            Collectors.joining(  
                System.lineSeparator(),  
                prefix: "",  
                System.lineSeparator()  
            )  
        );  
}
```

Java 12 – Indenting Strings

Java 8

```
public String indentation(String s, int indentationSize) {  
    return Stream.of(s.split(System.lineSeparator()))  
        .map(line -> {  
            StringBuilder stringBuilder = new StringBuilder();  
            for (int i = 0; i < indentationSize; i++) {  
                stringBuilder.append(' ');  
            }  
            stringBuilder.append(line);  
            return stringBuilder.toString();  
        }).collect(  
            Collectors.joining(  
                System.lineSeparator(),  
                prefix: "",  
                System.lineSeparator()  
            )  
        );  
}
```

Java 11

```
public String indentation(String s, int indentationSize) {  
    return s.lines()  
        .map(line -> " ".repeat(indentationSize) + line)  
        .collect(  
            Collectors.joining(  
                System.lineSeparator(),  
                prefix: "",  
                System.lineSeparator()  
            )  
        );  
}
```

Java 12

```
public String indentation(String s, int indentationSize) {  
    return s.indent(indentationSize);  
}
```

Java 14 – Enhanced Switch

Java 8

```
int value;

switch (dayOfWeek) {
    case MONDAY:
        value = 42;
        break;
    case TUESDAY:
        System.out.println("Today is your lucky day!");
        value = 100;
        break;
    case WEDNESDAY:
    case THURSDAY:
    case FRIDAY:
        value = 12;
        break;
    case SATURDAY:
    case SUNDAY:
        System.out.println("Woohoo! It's weekend!");
        value = 50;
        break;
    default:
        throw new IllegalStateException("Unexpected value: " + dayOfWeek);
}
```

Java 14 – Enhanced Switch

Java 8

```
int value;

switch (dayOfWeek) {
    case MONDAY:
        value = 42;
        break;
    case TUESDAY:
        System.out.println("Today is your lucky day!");
        value = 100;
        break;
    case WEDNESDAY:
    case THURSDAY:
    case FRIDAY:
        value = 12;
        break;
    case SATURDAY:
    case SUNDAY:
        System.out.println("Woohoo! It's weekend!");
        value = 50;
        break;
    default:
        throw new IllegalStateException("Unexpected value: " + dayOfWeek);
}
```

Java 14

```
int value;

value = switch (dayOfWeek) {
    case MONDAY -> 42;
    case TUESDAY -> {
        System.out.println("Today is your lucky day!");
        yield 100;
    }
    case WEDNESDAY, THURSDAY, FRIDAY -> 12;
    case SATURDAY, SUNDAY -> {
        System.out.println("Woohoo! It's weekend!");
        yield 50;
    }
};
```

Java 15 – Text Blocks

Java 15 – Text Blocks

Java 8

```
public String getSql() {  
    return new StringBuilder("SELECT")  
        .append(System.lineSeparator())  
        .append("    id,").append(System.lineSeparator())  
        .append("    name,").append(System.lineSeparator())  
        .append("    date_of_birth,").append(System.lineSeparator())  
        .append("    city").append(System.lineSeparator())  
        .append("FROM person").append(System.lineSeparator())  
        .append("WHERE").append(System.lineSeparator())  
        .append("    country = ? AND").append(System.lineSeparator())  
        .append("    active = ?").append(System.lineSeparator())  
        .toString();  
}
```

Java 15 – Text Blocks

Java 8

```
public String getSql() {
    return new StringBuilder("SELECT")
        .append(System.lineSeparator())
        .append("    id,").append(System.lineSeparator())
        .append("    name,").append(System.lineSeparator())
        .append("    date_of_birth,").append(System.lineSeparator())
        .append("    city").append(System.lineSeparator())
        .append("FROM person").append(System.lineSeparator())
        .append("WHERE").append(System.lineSeparator())
        .append("    country = ? AND").append(System.lineSeparator())
        .append("    active = ?").append(System.lineSeparator())
        .toString();
}
```

Java 15

```
public String getSql() {
    return """
        SELECT
            id,
            name,
            date_of_birth,
            city
        FROM person
        WHERE
            country = ? AND
            active = ?
        """;
}
```

Java 16 – Converting a Stream to an Unmodifiable List

Java 8

```
public void toList(Stream<String> stream) {  
    List<String> list = Collections.unmodifiableList(stream.collect(Collectors.toList()));  
}
```

Java 16 – Converting a Stream to an Unmodifiable List

Java 8

```
public void toList(Stream<String> stream) {  
    List<String> list = Collections.unmodifiableList(stream.collect(Collectors.toList()));  
}
```

Java 16

```
public void toList(Stream<String> stream) {  
    List<String> list = stream.toList();  
}
```

Java 16 – Records

Java 8

```
class Person {
    private final String name;
    private final LocalDate dateOfBirth;
    private final String country;

    Person(String name, LocalDate dateOfBirth, String country) {
        this.name = name;
        this.dateOfBirth = dateOfBirth;
        this.country = country;
    }

    public String getName() {
        return name;
    }

    public LocalDate getDateOfBirth() {
        return dateOfBirth;
    }

    public String getCountry() {
        return country;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (o == null || getClass() != o.getClass()) {
            return false;
        }
        Person person = (Person) o;
        return name.equals(person.name)
            && dateOfBirth.equals(person.dateOfBirth)
            && country.equals(person.country);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, dateOfBirth, country);
    }

    @Override
    public String toString() {
        return "Person[" +
            "name=" + name +
            ", dateOfBirth=" + dateOfBirth +
            ", country=" + country +
            ']';
    }
}
```

Java 16 – Records

Java 8

```
class Person {
    private final String name;
    private final LocalDate dateOfBirth;
    private final String country;

    Person(String name, LocalDate dateOfBirth, String country) {
        this.name = name;
        this.dateOfBirth = dateOfBirth;
        this.country = country;
    }

    public String getName() {
        return name;
    }

    public LocalDate getDateOfBirth() {
        return dateOfBirth;
    }

    public String getCountry() {
        return country;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (o == null || getClass() != o.getClass()) {
            return false;
        }
        Person person = (Person) o;
        return name.equals(person.name)
            && dateOfBirth.equals(person.dateOfBirth)
            && country.equals(person.country);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, dateOfBirth, country);
    }

    @Override
    public String toString() {
        return "Person[" +
            "name=" + name +
            ", dateOfBirth=" + dateOfBirth +
            ", country=" + country +
            ']';
    }
}
```

Project Lombok

@AllArgsConstructor

@ToString

@EqualsAndHashCode

static class Person {

@Getter private final String name;

@Getter private final LocalDate dateOfBirth;

@Getter private final String country;

}

Java 16 – Records

Java 8

```
class Person {
    private final String name;
    private final LocalDate dateOfBirth;
    private final String country;

    Person(String name, LocalDate dateOfBirth, String country) {
        this.name = name;
        this.dateOfBirth = dateOfBirth;
        this.country = country;
    }

    public String getName() {
        return name;
    }

    public LocalDate getDateOfBirth() {
        return dateOfBirth;
    }

    public String getCountry() {
        return country;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (o == null || getClass() != o.getClass()) {
            return false;
        }
        Person person = (Person) o;
        return name.equals(person.name)
            && dateOfBirth.equals(person.dateOfBirth)
            && country.equals(person.country);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, dateOfBirth, country);
    }

    @Override
    public String toString() {
        return "Person[" +
            "name=" + name +
            ", dateOfBirth=" + dateOfBirth +
            ", country=" + country +
            ']';
    }
}
```

Project Lombok

@AllArgsConstructor

@ToString

@EqualsAndHashCode

static class Person {

@Getter private final String name;

@Getter private final LocalDate dateOfBirth;

@Getter private final String country;

}

- Third-party library 🤔

Java 16 – Records

Java 8

```
class Person {
    private final String name;
    private final LocalDate dateOfBirth;
    private final String country;

    Person(String name, LocalDate dateOfBirth, String country) {
        this.name = name;
        this.dateOfBirth = dateOfBirth;
        this.country = country;
    }

    public String getName() {
        return name;
    }

    public LocalDate getDateOfBirth() {
        return dateOfBirth;
    }

    public String getCountry() {
        return country;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (o == null || getClass() != o.getClass()) {
            return false;
        }
        Person person = (Person) o;
        return name.equals(person.name)
            && dateOfBirth.equals(person.dateOfBirth)
            && country.equals(person.country);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, dateOfBirth, country);
    }

    @Override
    public String toString() {
        return "Person[" +
            "name=" + name +
            ", dateOfBirth=" + dateOfBirth +
            ", country=" + country +
            ']';
    }
}
```

Project Lombok

@AllArgsConstructor

@ToString

@EqualsAndHashCode

static class Person {

@Getter private final String name;

@Getter private final LocalDate dateOfBirth;

@Getter private final String country;

}

Java 16

```
record Person(String name, LocalDate dateOfBirth, String country) {
}
```

- Third-party library 🤔

- Pure Java 🙏

Java 16 – Pattern Matching for instanceof

Java 8

```
public void printLength(Object o) {  
    if (o instanceof String) {  
        System.out.println("The object has length = " + ((String) o).length());  
    } else {  
        System.out.println("It's not possible to get the object's length");  
    }  
}
```

Java 16 – Pattern Matching for instanceof

Java 8

```
public void printLength(Object o) {  
    if (o instanceof String) {  
        System.out.println("The object has length = " + ((String) o).length());  
    } else {  
        System.out.println("It's not possible to get the object's length");  
    }  
}
```

Java 16

```
public void printLength(Object o) {  
    if (o instanceof String s) {  
        System.out.println("The object has length = " + s.length());  
    } else {  
        System.out.println("It's not possible to get the object's length");  
    }  
}
```

Java 17 – Sealed classes and interfaces

```
sealed interface Vehicle permits Car, Motorcycle {  
    void run();  
}
```

```
class Truck implements Vehicle {  
    @Override  
    public void run() {  
        System.out.print  
    }  
}
```

Truck is not allowed in the sealed hierarchy

Add 'Truck' to permits list of a sealed class 'Vehicle' ↕ ↗ ↶ More actions... ⌘1

com.thegreatapi.newjdkfeatures.jdk17
interface SealedClasses.Vehicle

·jdk17

Java 17 – Sealed classes and interfaces

```
sealed interface Vehicle permits Car, Motorcycle {  
    void run();  
}
```

```
class Car implements Vehicle {
```

```
    @Override
```

```
    public
```

```
    run() {
```

```
    }
```

sealed, non-sealed or final modifiers expected

Make 'Car' final ↵ ↶ ↷ More actions... ⌘1

Java 17 – Sealed classes and interfaces

```
sealed interface Vehicle permits Car, Motorcycle {  
    void run();  
}
```

```
final class Car implements Vehicle {  
    @Override  
    public void run() {  
        System.out.println("Car noise");  
    }  
}
```

```
non-sealed class Motorcycle implements Vehicle {  
    @Override  
    public void run() {  
        System.out.println("Motorcycle noise");  
    }  
}
```

Java 17 – Supported By



QUARKUS



JAKARTA EE



MICROPROFILE™



spring

Start Using the New Features TODAY!

Clone the repo and try it:

<https://github.com/hbelmiro/newjdkfeatures>



Download OpenJDK 17
<https://adoptium.net>

<https://thegreatapi.com/>

http://twitter.com/helber_belmiro

<https://www.linkedin.com/in/hbelmiro>