Class and interface difference

Class	Interface
The keyword used to create a class is "class"	The keyword used to create an interface is "interface"
A class can be instantiated i.e., objects of a class can be created.	An Interface cannot be instantiated i.e. objects cannot be created.
Classes do not support multiple inheritance.	The interface supports multiple inheritance.
It can be inherited from another class.	It cannot inherit a class.
It can be inherited by another class using the keyword 'extends'.	It can be inherited by a class by using the keyword 'implements' and it can be inherited by an interface using the keyword 'extends'.
It can contain constructors.	It cannot contain constructors.
It cannot contain abstract methods.	It contains abstract methods only.

Why we used interfaces?

Here are some reasons why interfaces are used in Java:

- To achieve abstraction.

  Interfaces allow us to define a contract that classes must implement. This can be useful for hiding implementation details and providing a consistent interface to users of the class.

- To achieve polymorphism.

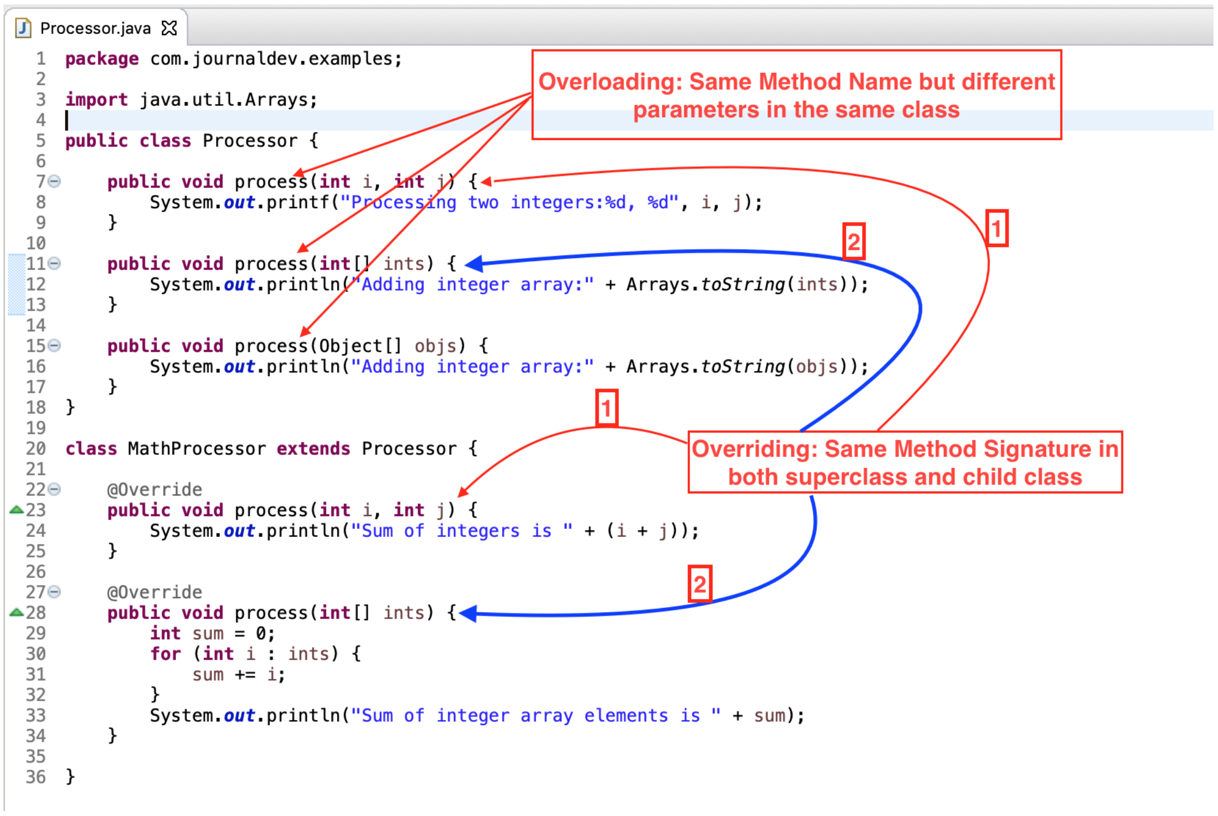
  Interfaces allow us to treat objects of different types in a similar way. For example, we can have a list of objects that implement the Drawable interface. We can then call the draw() method on any object in the list, without having to worry about the specific type of the object.

Why java is platform independent?

Java is platform-independent because it uses a "Write Once, Run Anywhere" approach. Java source code is compiled into bytecode, which is platform-neutral. This bytecode can be executed on any platform that has a Java Virtual Machine (JVM) compatible with that bytecode.

Comparing overriding and overloading

Overriding	**Overloading**
Implements “runtime polymorphism”	Implements “compile time polymorphism”
The method call is determined at runtime based on the object type	The method call is determined at compile time
Occurs between superclass and subclass	Occurs between the methods in the same class
Have the same signature (name and method arguments)	Have the same name, but the parameters are different
On error, the effect will be visible at runtime	On error, it can be caught at compile time


Processor.java
Overloading: Same Method Name but different parameters in the same class
Overriding: Same Method Signature in both superclass and child class
1
2
1
2

```java
package com.journaldev.examples;

import java.util.Arrays;

public class Processor {

    public void process(int i, int j) {
        System.out.printf("Processing two integers:%d, %d", i, j);
    }

    public void process(int[] ints) {
        System.out.println("Adding integer array:" + Arrays.toString(ints));
    }

    public void process(Object[] objs) {
        System.out.println("Adding integer array:" + Arrays.toString(objs));
    }
}

class MathProcessor extends Processor {

    @Override
    public void process(int i, int j) {
        System.out.println("Sum of integers is " + (i + j));
    }

    @Override
    public void process(int[] ints) {
        int sum = 0;
        for (int i : ints) {
            sum += i;
        }
        System.out.println("Sum of integer array elements is " + sum);
    }

}
```

Parameters	Java Application	Java Applet
Definition	Applications are just like a Java program that can be executed independently without using the web browser.	Applets are small Java programs that are designed to be included with the HTML web document. They require a Java-enabled web browser for execution.
main () method	The application program requires a main() method for its execution.	The applet does not require the main() method for its execution instead init() method is required.
Compilation	The “javac” command is used to compile application programs, which are then executed using the “java” command.	Applet programs are compiled with the “javac” command and run using either the “appletviewer” command or the web browser.
File access	Java application programs have full access to the local file system and network.	Applets don’t have local disk and network access.

Parameters	Java Application	Java Applet
Access level	Applications can access all kinds of resources available on the system.	Applets can only access browser-specific services. They don't have access to the local system.

Null pointer exception?

In Java, a NullPointerException occurs when a variable that is being accessed has not yet been assigned to an object, in other words, the variable is assigned as null.

```
String str = null;

System.out.println(str.length()); // throw null pointer exception
```

array Index Out Of Bounds Exception

An array Index Out Of Bounds Exception is thrown when a program attempts to access an element at an index that is outside the bounds of the array. This typically occurs when a program tries to access an element at an index that is less than 0 or greater than or equal to the length of the array.

```
int[] numbers = {1, 2, 3, 4, 5};
int index = -1;
try {
   int x = numbers[index]; // this will throw an ArrayIndexOutOfBoundsException
   System.out.println(x);
} catch (ArrayIndexOutOfBoundsException e) {
   System.out.println("Error: Index is out of bounds.");
}
```

Here are the access specifiers in Java with example:

Default - The default access modifier is also known as package-private access. This means that the class or member is only accessible within the same package. Here's an example of a class with default access:

Java

```
class DefaultAccessExample {
  int defaultVariable = 5;
  void defaultMethod() {
    System.out.println("Default method called.");
  }
}
```
Private - The private access modifier means that the class or member is only accessible within the same class. Here's an example of a class with private access:

```
Java
class PrivateAccessExample {
  private int privateVariable = 5;
  private void privateMethod() {
    System.out.println("Private method called.");
  }
}
```
Protected - The protected access modifier means that the class or member is accessible within the same package and by subclasses of the class. Here's an example of a class with protected access:

```
Java
class ProtectedAccessExample {
  protected int protectedVariable = 5;
  protected void protectedMethod() {
    System.out.println("Protected method called.");
  }
}
```
Public - The public access modifier means that the class or member is accessible from anywhere. Here's an example of a class with public access:

```
Java
class PublicAccessExample {
  public int publicVariable = 5;
  public void publicMethod() {
    System.out.println("Public method called.");
  }
}
```

why use packages?

Packages in Java are used to group related classes, interfaces, and sub-packages. We use Java packages to avoid naming conflicts since there can be two classes with the same name in two different packages.