Abstraction in Java

Abdul Ghafoor

Abstraction in Java

Data abstraction is the process of hiding certain details and showing only essential information to the user. Abstraction can be achieved with either

1. Abstract classes
2. Interfaces

Abstract keyword

- Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

Abstract Class

```
abstract class Animal {
public void sleep() {
System.out.println("Zzz");
}
 }
```

Abstract Method

```
abstract class Animal {
public abstract void animalSound();
 public void sleep() {
System.out.println("Zzz");
}
 }
```

Interfaces

- An interface is a fully abstract class. It includes a group of abstract methods (methods without a body).
- We use the **interface** keyword to create an interface in Java. For example:

```
interface Language {
  public void getType();

  public void getVersion();
}
```

- Language is an interface.
- It includes abstract methods: getType() and getVersion().

Implementing an Interface

- Like abstract classes, we cannot create objects of interfaces.
- To use an interface, other classes must implement it. We use the **implements** keyword to implement an interface.

Example 1: Java Interface

```
interface Polygon {
  void getArea(int length, int breadth);
}

// implement the Polygon interface
class Rectangle implements Polygon {

  // implementation of abstract method
  public void getArea(int length, int breadth) {
    System.out.println("The area of the rectangle is " + (length * breadth));
  }
}

class Main {
  public static void main(String[] args) {
    Rectangle r1 = new Rectangle();
    r1.getArea(5, 6);
  }
}
```

Example 2: Java Interface

```
// create an interface
interface Language {
  void getName(String name);
}

// class implements interface
class ProgrammingLanguage implements Language {

  // implementation of abstract method
  public void getName(String name) {
    System.out.println("Programming Language: " + name);
  }
}

class Main {
  public static void main(String[] args) {
    ProgrammingLanguage language = new ProgrammingLanguage();
    language.getName("Java");
  }
}
```

Implementing Multiple Interfaces

```
interface A {
  // members of A
}

interface B {
  // members of B
}

class C implements A, B {
  // abstract members of A
  // abstract members of B
}
```

Extending an Interface

- Similar to classes, interfaces can extend other interfaces. The **extends** keyword is used for extending interfaces. For example,

```
interface Line {
  // members of Line interface
}

// extending interface
interface Polygon extends Line {
  // members of Polygon interface
  // members of Line interface
}
```

Extending Multiple Interfaces

- An interface can extend multiple interfaces. For example,

```
interface A {
   ...
}
interface B {
   ...
}

interface C extends A, B {
   ...
}
```