Abstraction

▼ 1- In General

- In abstraction, only essential things are given to the user, details are hidden
- "Give the general things, ignore the details"
- There are two ways to achieve abstraction in java:
  - Abstract Class
  - Interface

▼ 2- Abstract Class

- A class declared with abstract keyword is an "Abstract Class"
- Abstract Class is also a "class". That means Abstract Classes have all the properties that non-abstract classes have.
- In this context an abstract class can have:
  - Variables
  - Constructors
  - Static Block
  - Instance Block
  - Instance Methods
  - Static Methods
  - Final Methods
  - Main Method
- Abstract classes can extend to other classes (abstract or non-abstract)
- Only difference is that:
  - Abstract classes can also have "abstract methods"

- We can not create object from an abstract class (can not be instantiated)

```
public abstract class MyAbstractClass {

   //variables
   public int x = 10;
   static double y = 10.5;

   //constructors
   public MyAbstractClass(){
      System.out.println("abstract classes can have constructors");
   }

   //static blocks
   static{
      System.out.println("abstract classes can have static blocks");
   }

   //instance blocks
   {
      System.out.println("abstract classes can have instance blocks");
   }

   //instance methods
   public void instanceMethod(){
      System.out.println("abstract classes can have instance methods");
   }

   //static methods
   public static void staticMethod(){
      System.out.println("abstract classes can have static methods");
   }

   //final method
   final public void finalMethod(){
      System.out.println("abstract classes can have final methods");
   }

   //abstract methods
   public abstract void  abstractMethod();

   //main method
   public static void main(String[] args) {
      System.out.println("abstract classes can have main method");
   }

}

//---------------------------------------------------------------------

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

```
      //✘ Compile Error: Abstract classes can not be instantiated
      MyAbstractClass object = new MyAbstractClass();
   }
}
```

▼ 3- Abstract Method

- A method which is:
  - Declared with abstract keyword
  - Does not have implementation (without curly braces)
- Abstract methods can only be declared in 'abstract classes' or 'interfaces'
- First concrete class (subclass of abstract class or interface) must implement all inherited abstract methods
- 'Method overriding' rules apply to implement abstract methods

```
public abstract class AbstractParent {

  //abstract method
  public abstract void abstractMethod();

}

//------------------------------------------------------------------------

class ConcreteChild extends AbstractParent{

  //First concrete class must implement all inherited abstract methods
  @Override
  public void abstractMethod(){
    System.out.println("abstract method is implemented");
  }

}
```

- An abstract class can extend to another abstract class or interface
- In that situation, it is optional for abstract class to implement abstract methods inherited from abstract super class

- But first concrete class must implement all the remaining abstract methods

```
public abstract class AbstractParent {
   public abstract void abstractMethod3();
   public abstract void abstractMethod4();
}

public interface MyInterface {
   void abstractMethod1();
   void abstractMethod2();
}

abstract class AbstractChild extends AbstractParent implements MyInterface{

   //it is optional to implement abstract methods
   //inherited from abstract super class
   @Override
   public void abstractMethod1(){
     System.out.println("abstract method 1 is implemented");
   }

   @Override
   public void abstractMethod3(){
     System.out.println("abstract method 3 is implemented");
   }

}

class FirstConcreteClass extends AbstractChild{

   //first concrete class must implement
   //all the remaining abstract methods
   @Override
   public void abstractMethod2(){
     System.out.println("abstract method 2 is implemented");
   }

   @Override
   public void abstractMethod4(){
     System.out.println("abstract method 4 is implemented");
   }

}
```

💡 Abstract classes may not be declared as "private" or "final". Because;

*Private classes may not be accessible for other classes, so we may not implement its abstract methods

*Final classes can not be inherited, so we can not implement its abstract methods

💡 Abstract methods may not be declared as "private" or "final". Because;

*Private methods are never inherited. So we can not implement an abstract private method

*Final methods are never overridden. So we can not implement a final abstract method

---

▼ 4- Interface

- It is the second way to achieve abstraction in Java
- It is similar to classes. But we use "interface" keyword instead of "class"
- 100% percent abstraction in Java is achieved using "interfaces"
- Before Java 8, only abstract methods are allowed in interfaces
- But since Java 8, "**public default"** and "**public static"** methods (non-abstract methods) are allowed in interfaces
- And since Java 9, "**private methods** "are also allowed in interfaces

```
public interface MyInterface {
    //some code
}
```

- What can an interface have?
  - public abstract methods

```
interface MyInterface{

  //first way
  public abstract void method1();

  //second way
  void method2(); //public abstract by default

}
```

  - public default methods (non-abstract method)

```
interface MyInterface{

  //first way
  public default void method1(){
    System.out.println("public default method");
  }

  //second way
  default void method2(){  //public default by default
    System.out.println("(public) default method");
  }

}
```

  - public static methods (non-abstract method)

```
interface MyInterface{

  //first way
  public static void method3(){
```

```
    System.out.println("public static method");
  }

  //second way
  static void method4(){  //public static by default
    System.out.println("(public) static method");
  }

}
```

- public static final variables

```
interface MyInterface{

   //first way
   public static final int x = 10;

   //second waY
   int y = 20;  //public static final by default

}
```

- What can not an interface have?
  - No constructor
  - No static block
  - No instance block
  - No access modifier other than "public"

- An interface can "extends" multiple interfaces
- A class can "implements" multiple interfaces
- A class can "extends" only one class
- An interface can not "extends" or "implements" a class