Assignment - 19
(Interface & Lambda Expression)

1. What is an Interface?

Ans: Interface is almost same as abstract class where all the methods are public and static by default, we don't have to declare it.
If you don't have idea about the implementation of the methods than you can use Interface where you just need to declare the methods.
Interface is used to create design or flow where we can reduce dependencies.

2. Which modifiers are allowed for methods in an Interface explain with an example?

Ans: Only **public** and **abstract** modifiers are allowed for methods in an Interface.

3. What is the use of Interface in Java?

Ans: There are many reasons to use Interface:-
a). An interface can be used to achieve full abstraction.
b). Using interfaces is the best way to expose our project's API to some other project.
c). Programmers use interfaces to customise features of software differently for different objects.
d). By using interface, we can achieve the functionality of multiple inheritance.

4. What is the difference between Abstract class and Interface?

Ans: **Abstract Class:**
a). Abstract class can have abstract and non - abstract methods.
b). Abstract class doesn't support multiple inheritance.
c). Abstract class can have **final, non-final, static, and non-static variables**.
d). The **abstract** keyword is use to declare abstract class.
e). An abstract class can extend another java class and implement multiple java interfaces.
f). An abstract class can be extended using **"extends"** keyword.
g). A java abstract class can have members like private, protected, etc.
h). example:
```
public abstract class shape{
        public abstract void draw(){
        }
}
```
i). Abstract class can provide the implementation of Interfaces.

Interfaces:
a). Interface can only have abstract methods, since Java 8 it can have **default** and **static** methods.
b). Interfaces supports multiple Inheritance.
c). Interfaces only have **static** and **final** variables.
d). The **interface** keyword is used to declare the Interface.
e). An Interface can extend another Interface only.
f). An Interface can be implemented using "**implements"** keyword.
g). Members of a Java Interface are public by default.
h). example:
```
public interface Drawable{
        void draw();{
        }
}
```
i). Interface can't provide the implementation of Abstract class.

5. What is Lambda Expression in Java 8?

Ans: As its name suggests it's an expression which allows you to write more succinct code in Java 8. For example (a, b) -> a + b is a lambda expression (look for that arrow ->).
Which is equal to following code:
```
public int value(int a, int b){
return a + b;
}
```

It's also called an anonymous function because you are essentially writing the code you write in function but
without name.

6. Can you pass lambda expressions to a method? When?

Ans: Yes, you can pass a lambda expression to a method provided it is expecting a functional interface. For example, if a method is accepting a Runnable, Comparable or Comparator then you can pass a lambda expression to it because all these are functional interfaces in Java 8.

7. What is the functional interface in Java 8?

Ans: A functional interface in Java 8 is an interface with a single abstract method. For example, Comparator which has just one abstract method called compare() or Runnable which has just one abstract method called run(). There are many more general purpose functional interfaces introduced in JDK on java.util.function package. They are also annotated with @FunctionalInterface but that's optional.

8. What is the benefit of lambda expressions in Java 8?

Ans: The main benefit of lambda expression in Java 8 is that now it's easier to pass a code block to a method.
Earlier, the only way to do this was wrapping the code inside an Anonymous class, which requires a lot of boilerplate code.

9. Is it mandatory for a lambda expression to have parameters?

Ans: No, it's not mandatory for a lambda expression to have parameters, you can define a lambda expression
without parameters as shown below:
() -> System.out.println("lambdas without parameter");
You can pass this code to any method which accepts a functional interface.