Assignment Questions 4

Q1. Write a program to show Interface Example in java?

```
interface printable{
void print();
}
class A6 implements printable{
public void print(){System.out.println("Hello");}

public static void main(String args[]){
A6 obj = new A6();
obj.print();
 }
}
```

Q2. Write a program a Program with 2 concrete method and 2 abstract method in java ?

```
// Abstract class example
abstract class AbstractExample {

   // Abstract method
   abstract void display();

   // Concrete method
   void show()
   {
      System.out.println("Concrete method of abstract class");
   }
}

// Subclass of abstract class
class SubClass extends AbstractExample {

   // Implementing the abstract method
   void display()
   {
      System.out.println("Abstract method implemented");
```

```
    }
}

/**
 * Main class
 */
public class AbstractClass{

    public static void main(String args[])
    {
        // Creating an object of the subclass
        SubClass obj = new SubClass();

        // Calling the abstract method
        obj.display();

        // Calling the concrete method
        obj.show();
    }
}
```

Q3. Write a program to show the use of functional interface in java?

```
class Test {
    public static void main(String args[])
    {
        // create anonymous inner class object
        new Thread(new Runnable() {
            @Override public void run()
            {
                System.out.println("New thread created");
            }
        }).start();
    }
}
```

Q4. What is an interface in Java?

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

Q5. What is the use of interface in Java?

- **Provides communication** − One of the uses of the interface is to provide communication. Through interface you can specify how you want the methods and fields of a particular type.
- **Multiple inheritance** − Java doesn't support multiple inheritance, using interfaces you can achieve multiple inheritance −

Q6. What is the lambda expression of Java 8?

Lambda Expressions were added in Java 8.
A lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.

Q7. Can you pass lambda expressions to a method? When?

A lambda expression can be passed to a method as an argument. The declaration of such a method must contain a reference to the corresponding functional interface. This reference is obtained as a parameter of the method that processes the lambda expression.
There are two ways to pass a lambda expression to a method:

- passing the lambda expression directly. This method works well for lambda expressions with few operators;
- passing a reference to the functional interface that is associated with the lambda expression. In this case, a reference to the interface is pre-declared. The lambda expression code is then assigned to this reference. This technique is appropriate when the lambda expression becomes too long to be embedded in a method call.

Q8. What is the functional interface in Java 8?

An Interface that contains exactly one abstract method is known as functional interface. It can have any number of default, static methods but can contain only one abstract method. It can also declare methods of object class.
Functional Interface is also known as Single Abstract Method Interfaces or SAM Interfaces.

Q9. What is the benefit of lambda expressions in Java 8?

- **Fewer Lines of Code** − One of the most benefits of a lambda expression is to **reduce the amount of code**. We know that lambda expressions can be used only with a **functional interface**. For instance, **Runnable** is a functional interface, so we can easily apply lambda expressions.
- **Sequential and Parallel execution support by passing behavior as an argument in methods** − By using **Stream API** in Java 8, the functions are passed to collection methods. Now it is the responsibility of collection for processing the elements either in a sequential or parallel manner.
- **Higher Efficiency** − By using **Stream API** and **lambda expressions**, we can achieve higher efficiency (**parallel execution**) in case of bulk operations on collections. Also, lambda expression helps in achieving the internal iteration of collections rather than external iteration.

Q10. Is it mandatory for a lambda expression to have parameters?

No need to declare the type of a parameter. The compiler can inference the same from the value of the parameter.