

OOPM Assignment - 2

Yash Sarang D6AD/47

Q1. \rightarrow The ~~x~~ exception handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow is maintained.

Exception means abnormal condition. In Java, exception is an event that disrupts the normal flow of the program.

It is an object which is thrown at runtime.

Q2. \rightarrow We need to synchronize the shared resources to ensure that at a time only one thread is able to access the shared resource.

If an object is shared by multiple threads then there is need of synchronization in order to avoid the objects state to be getting corrupted.

Example:-

Non Synchronized Code:

```
class Table {  
    void printTable(int n) {           //method not synchronized  
        for (int i = 1; i <= 5; i++) {  
            System.out.println(n * i);  
            try {  
                Thread.sleep(400);  
            } catch (Exception e) {  
                System.out.println(e);  
            }  
        }  
    }  
}  
  
class MyThread1 extends Thread {  
    Table t;  
    MyThread1(Table t) {  
        this.t = t;  
    }  
    public void run() {  
        t.printTable(5);  
    }  
}  
  
class MyThread2 extends Thread {  
    Table t;  
    MyThread2(Table t) {  
        this.t = t;  
    }  
    public void run() {  
        t.printTable(100);  
    }  
}  
  
class TestSynchronization1 {  
    public static void main(String args[]) {  
        Table obj = new Table(); //only one object  
        MyThread1 t1 = new MyThread1(obj);  
        MyThread2 t2 = new MyThread2(obj);  
        t1.start();  
        t2.start();  
    }  
}
```

OUTPUT:

```
5
100
10
200
15
300
20
400
25
500
```

Synchronized Code: -

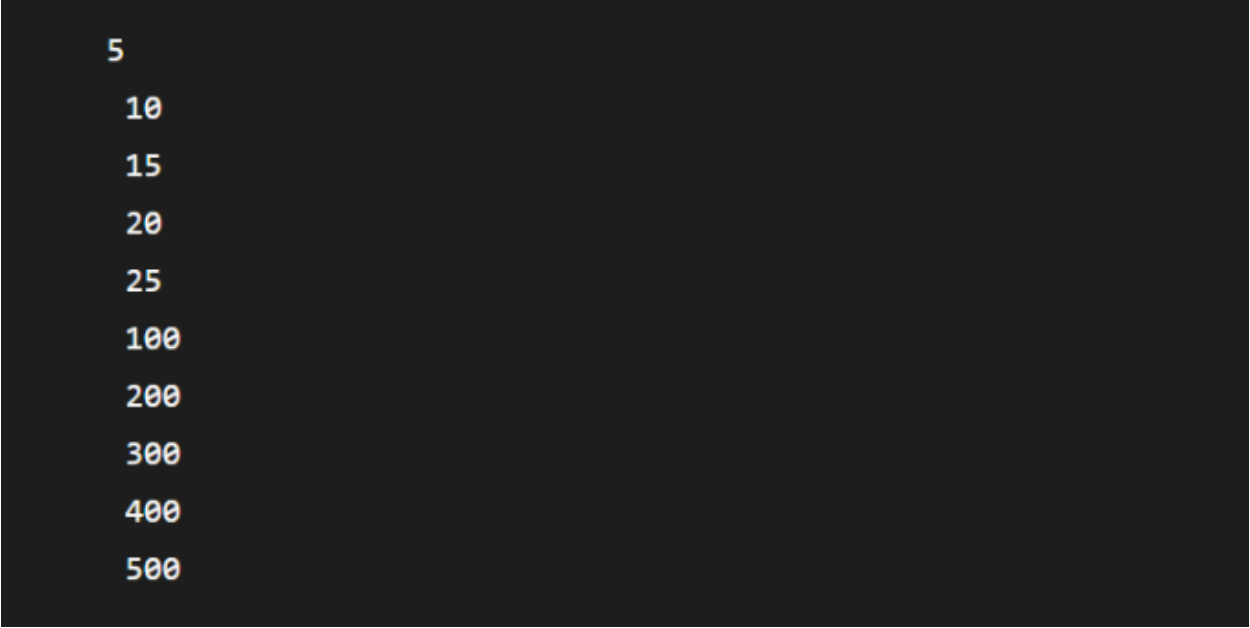
```
class Table {
    synchronized void printTable(int n) { //synchronized method
        for (int i = 1; i <= 5; i++) {
            System.out.println(n * i);
            try {
                Thread.sleep(400);
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }
}

class MyThread1 extends Thread {
    Table t;
    MyThread1(Table t) {
        this.t = t;
    }
    public void run() {
        t.printTable(5);
    }
}

class MyThread2 extends Thread {
    Table t;
    MyThread2(Table t) {
        this.t = t;
    }
}
```

```
    public void run() {  
        t.printTable(100);  
    }  
}  
public class TestSynchronization2 {  
    public static void main(String args[]) {  
        Table obj = new Table(); //only one object  
        MyThread1 t1 = new MyThread1(obj);  
        MyThread2 t2 = new MyThread2(obj);  
        t1.start();  
        t2.start();  
    }  
}
```

OUTPUT:



```
5  
10  
15  
20  
25  
100  
200  
300  
400  
500
```


Q3.

We perform event handling in Swing & AWT, we can also do it in Applets.

GUI contains components of the user interface, they generate user interactions using a key on the keyboard or clicking a mouse button.

When an applet is designed, all these events are captured & the appropriate actions are performed in response to each of the events provided.

for eg. we must implement interfaces related to the kind of event that is generated by our program.
for eg. if our program has a button in it & in order to listen a button in its click event, we must implement action listener interface & implement its method `actionPerformed()`.

Q4. Applets:-

An applet is a program written in Java programming language that can be included in an HTML page. When you use the Java technology enabled browser to view a page that contains an applet, the applet's code is transferred to your system & executed by the browser's JVM.

SWING:-

Swing is a widget tool for Java. It is a part of SUN JFC - an API for providing a graphical user interface.

It was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit.

It provides a native look & feel that emulates the look & feel of several platforms.

Q5. JDBC Driver is a software component that enables ~~for~~ a java application to interact with the database.

① JDBC - ODBC bridge driver.

Uses the ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into ODBC function calls.

② Native API driver

Uses the client side libraries of the database. It converts JDBC method calls into native calls of the database API.

③ Network protocol driver.

It uses middleware that connects JDBC calls directly or indirectly into the vendor-specific database protocol.

④ Thin driver.

Converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver.

The steps are ::

- ① Import JDBC packages.
- ② Load & register JDBC packages.
- ③ Open a connection to database
- ④ Create a statement object to perform a query.
- ⑤ Execute a statement object & run query.
- ⑥ Process the result set.
- ⑦ Close the result set & statement object.
- ⑧ Close the connection.