```
/*
Name          : Deepanshu Gupta
Section       : AI & ML
Roll Number   : 10
1. Create a three-level hierarchy of exceptions. Now create a base-class A with a method that
throws an exception at the base of your hierarchy. Inherit B from A and override the method
so it throws an exception at level two of your hierarchy. Repeat by inheriting class C from B.
In main( ), create a C and upcast it to A, then call the method.
*/


class LevelOneException extends Exception{
   public LevelOneException(){
      System.out.println("Level One Exception");
   }
}
class LevelTwoException extends LevelOneException{
   public LevelTwoException(){
      System.out.println("Level Two Exception");
   }
}
class LevelThreeException extends LevelTwoException{
   public LevelThreeException(){
      System.out.println("Level Three Exception");
   }
}
class A{
   public void method() throws LevelOneException{
      throw new LevelOneException();
   }
}
class B extends A{
   public void method() throws LevelTwoException{
```

```
        throw new LevelTwoException();
    }
}
class C extends B{
    public void method() throws LevelThreeException{
        throw new LevelThreeException();
    }
}
public class Main{
    public static void main(String[] args){
        A obj=new C();
        try{
            obj.method();
        }
        catch (LevelOneException e){
            System.out.println("Caught Level One Exception");
        }
    }
}
```

*****OUTPUT*****

```
Level One Exception
Level Two Exception
Level Three Exception
Caught Level One Exception
```

```
/*
Name          : Deepanshu Gupta
Section       : AI & ML
Roll Number   : 10
2. WAP to show that how producer-consumer problem can be solved using Inter thread communication in Java.
*/


class Customer{
   int amt=10000;
   synchronized void withdraw(int money){
      System.out.println("Withdraw");
      if(amt<money){
         System.out.println("Less balance");
         try{
            wait();
         } catch(Exception e){}
      }
      amt=amt-money;
      System.out.println("Withdraw complete");
      System.out.println("Updated balance="+amt);
   }
   synchronized void deposit(int money){
      System.out.println("Deposit");
      amt=amt+money;
      System.out.println("Deposit complete");
      notify();
   }
}
class T1 extends Thread{
   Customer c;
   T1(Customer c){
```

```
        this.c=c;
    }
    public void run(){
        c.withdraw(15000);
    }
}
class T2 extends Thread{
    Customer c;
    T2(Customer c){
        this.c=c;
    }
    public void run(){
        c.deposit(10000);
    }
}
public class Main {
    public static void main(String[] args) {
        Customer c=new Customer();
        T1 ob1=new T1(c);
        T2 ob2=new T2(c);
        ob1.start();
        ob2.start();
    }
}
```

*****OUTPUT*****

```
Withdraw
Less balance
Deposit
Deposit complete
Withdraw complete
Updated balance=5000
```

```
/*
Name          : Deepanshu Gupta
Section       : AI & ML
Roll Number   : 10
3. WAP in which there is one class Table, which is having a function to print table of any number(number should be passed as a function parameter).Create two threads First and Second and both threads are printing table of two different numbers. Create one more class as Final class in which run method of both the threads will be called.
*/


class Table extends Thread{
    synchronized void printTable(int n){
        for(int i=1; i<=5; i++){
            System.out.println(n*i);
            try{
                Thread.sleep(400);
            }catch(Exception e){
                System.out.println(e);
            }
        }
    }
    Thread t1=new Thread(){
        public void run(){
            printTable(5);
        }
    };
    Thread t2=new Thread(){
        public void run(){
            printTable(100);
        }
    };
}
public class Main {
```

```
    public static void main(String[] args) {
        Table obj=new Table();
        obj.t1.start();
        obj.t2.start();
    }
}
```

*****OUTPUT*****

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

```
/*
Name          : Deepanshu Gupta
Section       : AI & ML
Roll Number   : 10
4. WAP to show the concept of deadlock condition between multiple threads when they are sharing common resources also write code that how this deadlock can be resolved.
*/


//DEADLOCK EXAMPLE
public class DeadlockExample{
   private static Object r1=new Object();
   private static Object r2=new Object();
   public static void main(String[] args){
      Thread t1 = new Thread(()->{
         synchronized(r1){
            System.out.println("Thread 1: Resource 1");
            try{
               Thread.sleep(100);
            }
            catch(InterruptedException e){
               e.printStackTrace();
            }
            synchronized(r2){
               System.out.println("Thread 1: Resource 2");
            }
         }
      });
      Thread t2 = new Thread(()->{
         synchronized(r2){
            System.out.println("Thread 2: Resource 2");
            try{
               Thread.sleep(100);
```

```
            } catch(InterruptedException e){
                e.printStackTrace();
            }
            synchronized(r1){
                System.out.println("Thread 2: Resource 1");
            }
        }
    });
    t1.start();
    t2.start();
  }
}
```

*****OUTPUT*****

```
Thread 1: Resource 1
Thread 2: Resource 2
…….
```

```
//DEADLOCK RESOLUTION EXAMPLE
public class DeadlockResolutionExample{
  private static Object r1=new Object();
  private static Object r2=new Object();
  public static void main(String[] args) {
    Thread t1=new Thread(()->{
      synchronized(r1) {
        System.out.println("Thread 1: Resource 1");
        try {
          Thread.sleep(100);
        }
        catch(InterruptedException e){
          e.printStackTrace();
        }
        synchronized(r2){
```

```
                System.out.println("Thread 1: Resource 2");
            }
        }
    });
    Thread t2=new Thread(()->{
        synchronized(r1){
            System.out.println("Thread 2: Resource 1");
            try {
                Thread.sleep(100);
            }
            catch (InterruptedException e){
                e.printStackTrace();
            }
            synchronized (r2) {
                System.out.println("Thread 2: Resource 2");
            }
        }
    });
    t1.start();
    t2.start();
  }
}
```

*****OUTPUT*****

```
Thread 1: Resource 1
Thread 1: Resource 2
Thread 2: Resource 1
Thread 2: Resource 2
```