# Java Assignment2

## //ClassNotFoundException and NoClassDefFoundError

```
public class ClassNotFoundException {
        public static void main(String[] args)
        {
            try
            {
              Class.forName("oracle.jdbc.driver.OracleDriver");
            }catch (ClassNotFoundException e)
            {
                e.printStackTrace();
            }
        }
    }
}


    class A
    {
        // some code
    }
//it will raise NoClassDefFoundError, if we delete the .A classfile after compilation of code
    public class NoClassDefFoundError {
        public static void main(String[] args)
        {
            A a = new A();
        }
    }
```

/// CloneByUsingClonable

```
class CloneByUsingClonable implements Cloneable{
        int rollno;
        String name;
        CloneByUsingClonable(int rollno,String name){
            this.rollno=rollno;
            this.name=name;
        }
        public Object clone()throws CloneNotSupportedException{
            return super.clone();
        }
        public static void main(String args[]){
            try{
                CloneByUsingClonable s1=new CloneByUsingClonable(101,"amit");
                CloneByUsingClonable s2=(CloneByUsingClonable)s1.clone();
                System.out.println(s1.rollno+" "+s1.name);
                System.out.println(s2.rollno+" "+s2.name);
            }catch(CloneNotSupportedException c){}
        }
    }
```

```
//CloneByUsingCopyConstructor

public class CloneByUsingCopyConstructor
    {
        public int x;
        public int y;
        //Constructor with 2 parameters
        public CloneByUsingCopyConstructor(int x, int y)
        {
            super();
            this.x = x;
            this.y = y;
        }
        //Copy Constructor
        public CloneByUsingCopyConstructor(CloneByUsingCopyConstructor p)
        {
            this.x = p.x;
            this.y = p.y;
        }
        public static void main(String[] args)
        {
            CloneByUsingCopyConstructor p1 = new CloneByUsingCopyConstructor(1,2);
            CloneByUsingCopyConstructor p2 =  new CloneByUsingCopyConstructor(p1);
            System.out.println(p1.x + " " + p1.y); // prints "1 2"
            System.out.println(p2.x + " " + p2.y); // prints "1 2"
            p2.x = 3;
            p2.y = 4;
            System.out.println(p1.x + " " + p1.y); // prints "1 2"
            System.out.println(p2.x + " " + p2.y); // prints "3 4"
        }
    }
```

// runtime input

```
import java.util.Scanner;
public class Conditional8 {
    public static void main(String[] a) {
        Scanner s = new Scanner(System.in);
        boolean flag = true;
        while (flag) {
            System.out.println("enter string and check whether first and last char is
equal or not till done");
            String s1 = s.nextLine();
            if (s1.equals("done")) {
                flag = false;
            }
                else if (s1.charAt(0) == s1.charAt(s1.length() - 1)) {
                    System.out.println(" char are equal");
                }
            }
        }
    }
```

//inheritance operation

```java
class Parent extends Grandparent {
    {
        System.out.println("instance - parent");
    }
    public Parent() {
        System.out.println("constructor - parent");
    }
    static {
        System.out.println("static - parent");
    }
}
class Grandparent {
    static {
        System.out.println("static - grandparent");
    }
    {
        System.out.println("instance - grandparent");
    }
    public Grandparent() {
        System.out.println("constructor - grandparent");
    }
}
class Child extends Parent {
    public Child() {
        System.out.println("constructor - child");
    }
    static {
        System.out.println("static - child");
    }
    {
        System.out.println("instance - child");
    }
}
public class InheritanceOperation {
    public static void main(String[] a){
        Grandparent gp=new Child();
    }
}
```

//multicatch

```java
import java.util.Scanner;
public class MultiCatch {
        public static void main(String[] args)
        {
            int dividend, divisor, quotient;
            Scanner console = new Scanner(System.in);
            try
            {
                System.out.print("Enter the dividend : ");
                dividend = console.nextInt();
                System.out.print("Enter the divisor : ");
                divisor = console.nextInt();
                quotient = dividend / divisor;
                System.out.println("Quotient = " + quotient);
            }
            catch (ArithmeticException ex)
            {
                System.out.println("Exception: " + ex.toString());
            }
            catch (Exception ex)
            {
                System.out.println("Exception: " + ex.toString());
            }
            finally{
                System.out.println("finally block");
            }
        }
    }
```

//Furniture

```
interface Furniture{
    public void stressTest();
    public void fireTest();
}
class WoodenChair implements Furniture{
    WoodenChair(){
        System.out.println("wooden chair");
    }
    public void stressTest(){
    System.out.println(" stress Test passed");
    }
    public void fireTest(){
        System.out.println(" fire test not passed");
    }
}
class WoodenTable implements Furniture{
    WoodenTable(){
        System.out.println("Wooden table");
    }
    public void stressTest(){
        System.out.println(" stress Test passed");
    }
    public void fireTest(){
        System.out.println(" fire test not passed");
    }
}
class MetalChair implements Furniture{
    MetalChair(){
        System.out.println("Metal Chair");
    }
    public void stressTest(){
        System.out.println(" stress Test passed");
    }
    public void fireTest(){
        System.out.println(" fire test passed");
    }
}
class MetalTable implements Furniture{
    MetalTable(){
        System.out.println("Metal Table");
    }
    public void stressTest(){
        System.out.println(" stress Test passed");
    }
    public void fireTest(){
        System.out.println(" fire test not passed");
    }
}
public class OFurniture {
    public static void main(String[] af){
        Furniture f1=new WoodenChair();
        f1.fireTest();
        f1.stressTest();
        Furniture f2=new WoodenChair();
        f2.fireTest();
        f2.stressTest();
        Furniture f3=new MetalChair();
        f3.fireTest();
        f3.stressTest();
        Furniture f4=new MetalTable();
        f4.fireTest();
       f4.stressTest();
    }
}
```

```
// Singleton
class App{
    private static App app;
    public String str;
    private App(){
     str="hello its me aditya";
    }
    public static App getInstance(){
        if(app==null)
        {
            app=new App();
        }
        return app;
    }
}
public class Singleton{
    public static void main(String[] ag){
        App app=App.getInstance();
        System.out.println(app.str);
        App app1=App.getInstance();
        System.out.println(app1.str);
        System.out.println(app);
        System.out.println(app1);
    }
}
```

// Sorting string

```
public class SotingString {
    public static void getSortedString(String str){
        char[] ch=str.toCharArray();
        int len=ch.length;
        char tem;
        for(int i=0;i<len;i++) {
            for (int j = 0; j < len; j++) {
                if (ch[i] <= ch[j]) {
                    tem = ch[j];
                    ch[j] = ch[i];
                    ch[i] = tem;
                }
            }
        }
        String s=new String(ch);
        System.out.println(s);
    }
    public static void main(String[] ag){
        getSortedString("aditya");
    }
}
```

//Supress stack trace in exception handling

```
import java.lang.RuntimeException;
class SponsoredException extends RuntimeException {
         @Override
         public synchronized Throwable fillInStackTrace() {
             setStackTrace(new StackTraceElement[]{
                      new StackTraceElement("ADVERTISEMENT", "   If you don't   ", null,
0),
                      new StackTraceElement("ADVERTISEMENT", " want to see this ", null,
0),
                      new StackTraceElement("ADVERTISEMENT", "     exception    ", null,
0),
                      new StackTraceElement("ADVERTISEMENT", "    please  buy   ", null,
0),
                      new StackTraceElement("ADVERTISEMENT", "   full  version  ", null,
0),
                      new StackTraceElement("ADVERTISEMENT", "  of  the program ", null,
0)
             });
             return this;
         }
    }
    public class SupressStackTrace extends RuntimeException {
         public SupressStackTrace() {
             super("Catch me if you can");
         }
         @Override
         public synchronized Throwable fillInStackTrace() {
             return this;
         }
    }
```

//time convertor

```
public class TimeConverter
    {
         // function convert second into day
         // hours, minutes and seconds
       public  static void ConvertSectoDay(int n)
         {
             int day = n / (24 * 3600);
             n = n % (24 * 3600);
             int hour = n / 3600;
             n %= 3600;
             int minutes = n / 60 ;
             n %= 60;
             int seconds = n;
             System.out.println( day + " " + "days " + hour
                      + " " + "hours " + minutes + " "
                      + "minutes " + seconds + " "
                      + "seconds ");
         }
         public static void main (String[] args)
         {
             // Given n is in seconds
             int n = 129600;
             ConvertSectoDay(n);
         }
    }
```

//while operation

```
public class WhileOperation {
    public static void main(String[] ag){
        int s = 0;
        int t = 1;
    //  for (int i = 0; i < 10; i++)
        int i=0;
        while(i<10)
        {
            s = s + i;
          //  for (int j = i;j > 0;j--)
            int j=i;
            while(j>0)
            {
                t = t * (j - i);
                j--;
            }
            s = s * t;
            System.out.println("T is " + t);
            i++;
        }
        System.out.println("S is " + s);
    }
}
```

//Cafeteria

```
import java.util.Scanner;
abstract class CoffeeShop {
    abstract int requestService();
    abstract void processOrder();
    abstract void deliverService();
}
class Cafeteria extends CoffeeShop {
    //private static Question10 st = new Question10();
    private static int token_number=1000;
    private double order_amount;
    private boolean token_status;
    private  boolean payment_status;
    private int[] order_queue;
    private int[] ready_queue;
    private final int MAX_ORDER;
    private Scanner In;
    private static int o_count=-1;
    public Cafeteria() {
        MAX_ORDER=10;
        token_status=false;
        payment_status=false;
        order_queue=new int[MAX_ORDER];
        ready_queue=new int[MAX_ORDER];
        In=new Scanner(System.in);
    }
   /*public static Question10 getInstance() {
       return st;
   }*/
    @Override
    public int requestService() {
        o_count++;
        token_number++;
        System.out.println("[Pay Bill] Enter amount: ");
        order_amount=In.nextLong();
        if(order_amount>=0) {
            order_queue[o_count]=token_number;
            System.out.println("Order id "+order_queue[o_count]+" of amount 
"+order_amount +" is in order queue, please wait.");
            return  order_queue[o_count];
        } else {
            System.out.println("Amount not paid, try again.");
            return 0;
        }
    }
    @Override
    public void processOrder() {
        if(order_queue.length>0) {
            for(int i=0;i<order_queue.length;i++) {
                if(order_queue[i]==0) {
                    break;
                } else {
                    System.out.println("Order id " + order_queue[i] + " is ready.");
                    ready_queue[i] = order_queue[i];
                }
            }
            order_queue=null;
        }
    }
    @Override
    public void deliverService() {
        if(ready_queue.length>0) {
            for(int i=0;i<ready_queue.length;i++) {
                if(ready_queue[i]==0) {
                    break;
                } else {
                    System.out.println("Order id " + ready_queue[i] + " is 
completed.");
                }
            }
            ready_queue=null;
        }
    }
}
public class OCafeteria {
    public static void main(String[] args) {
        Cafeteria Q10=new Cafeteria();
        int tNumber=Q10.requestService();
        System.out.println("Your token number(order number) is: "+tNumber);
        Q10.processOrder();
        Q10.deliverService();
    }
}
```

//Library management

```
interface Library{
    public static void getName(){
        System.out.println("Public library");
    }
   public void type();
}
abstract class Book implements Library{
  abstract   void getBookCollection();
}
class LibraryManagement extends Book{
    public void type(){
        System.out.println("Monthly payment");
    }
    public void getBookCollection(){
        System.out.println("1 million books");
    }
    public static void main(String[] ag){
        Library library;
        library=new LibraryManagement();
        library.type();
        Library.getName();
    }
}
```