Introduction to Java 2 Exercise

1. Create Java classes having suitable attributes for Library management system.Use OOPs concepts in your design.Also try to use interfaces and abstract classes.

Sol –

```
public enum BookFormat {
 hardcover,paperback,ebook,newspaper,magazine
}
public enum BookStatus {
  available, reserved,loaned,lost
}
public enum ReservationStatus{
  aiting,pending,canceled,none
}
public enum AccountStatus{
  active, closed, canceled,blanklisted,none
}
public class Address {
  private String streetAddress;
  private String city;
  private String state;
  private String PinCode;
  private String country;
}
public class Person {
  private String name;
  private Address address;
  private String email;
  private String phone;
}
```

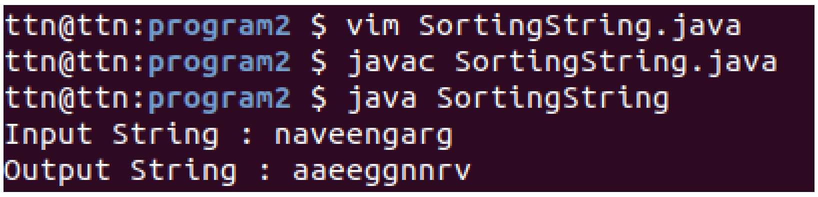
```
public class Constants {
  public static final int MAX_BOOKS_ISSUED_TO_A_USER = 5;
  public static final int MAX_LENDING_DAYS = 10;
}

//BookReservation, BookLending, and Fine
public class BookReservation {
        private Date creationDate;
        private ReservationStatus status;
        private String bookItemBarcode;
        private String memberId;
}

public class BookLending {
        private Date creationDate;
        private Date dueDate;
        private Date returnDate;
        private String bookItemBarcode;
        private String memberId;
}

public class Fine {
        private Date creationDate;
        private String memberId;
        public static void collectFine(String memberId, long days) {}
}

//Account, Member, and Librarian
public abstract class Account {
        private String id;
        private String password;
        private AccountStatus status;
        private Person person;
         public boolean resetPassword();
}

public class Librarian extends Account {
        public boolean addBookItem(BookItem bookItem);
        public boolean blockMember(Member member);
        public boolean unBlockMember(Member member);
}

public class Member extends Account {
        private Date dateOfMembership;
        private int totalBooksCheckedout;
        public int getTotalBooksCheckedout();
        public boolean reserveBookItem(BookItem bookItem);
        private void incrementTotalBooksCheckedout();
}
//BookItem
public abstract class Book {
        private String title;
        private String subject;
        private String publisher;
        private String language;
        private int numberOfPages;
        private List<Author> authors;
}

public class BookItem extends Book {
        private String barcode;
        private Date borrowed;
        private Date dueDate;
        private double price;
        private BookFormat format;
        private BookStatus status;
        private Date dateOfPurchase;
        private Date publicationDate;
        public boolean checkout(String memberId) {
                //code
        }
}
//Search interface
public interface Search {
        public List<Book> searchByTitle(String title);
        public List<Book> searchByAuthor(String author);
        public List<Book> searchBySubject(String subject);
}

public class Catalog implements Search {
        private HashMap<String, List<Book>> bookTitles;
        private HashMap<String, List<Book>> bookAuthors;
        private HashMap<String, List<Book>> bookSubjects;

        public List<Book> searchByTitle(String query) {
                 // return all books
                return bookTitles.get(query);
        }

        public List<Book> searchByAuthor(String query) {
                // return all books
                return bookAuthors.get(query);
  }
```

```
}
public class MainClass{
        public static void main(String arg[])
        {
                //code
        }
}
```

2. WAP to sorting string without using string Methods?

Sol – Program Files Folder name – program2

```
import java.util.Arrays;
public class SortingString
{
    public static String temp(String inputString)
    {
        char tempArray[] = inputString.toCharArray();
        Arrays.sort(tempArray);
        return new String(tempArray);
    }

    public static void main(String[] args)
    {
        String inputString = "naveengarg";
        char tempArray[] = inputString.toCharArray();
        String outputString = temp(inputString);
        System.out.println("Input String : " + inputString);
        System.out.println("Output String : " + outputString);
    }
}
```



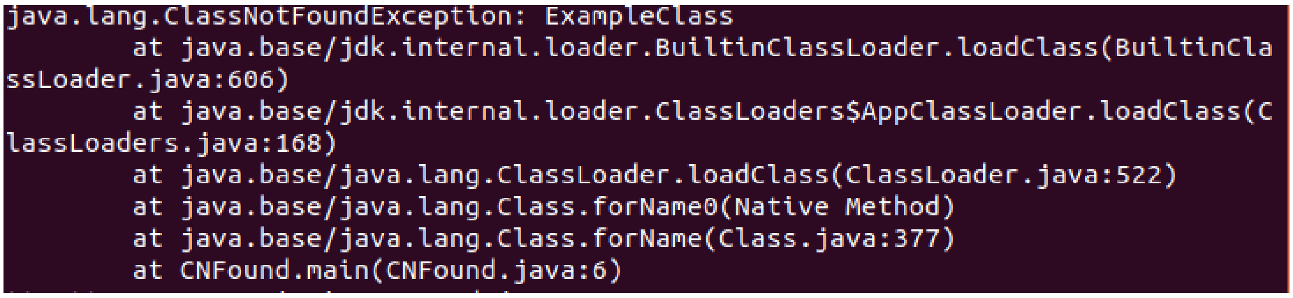
```
ttn@ttn:program2 $ vim SortingString.java
ttn@ttn:program2 $ javac SortingString.java
ttn@ttn:program2 $ java SortingString
Input String : naveengarg
Output String : aaeeggnnrv
```

3. WAP to produce NoClassDefFoundError and ClassNotFoundException exception.

Sol – Program Files Folder name – program3

ClassNotFoundException - ClassNotFoundException is a runtime exception that is thrown when an application tries to load a class at runtime using the Class.forName() or loadClass() or findSystemClass() methods ,and the class with specified name are not found in the classpath

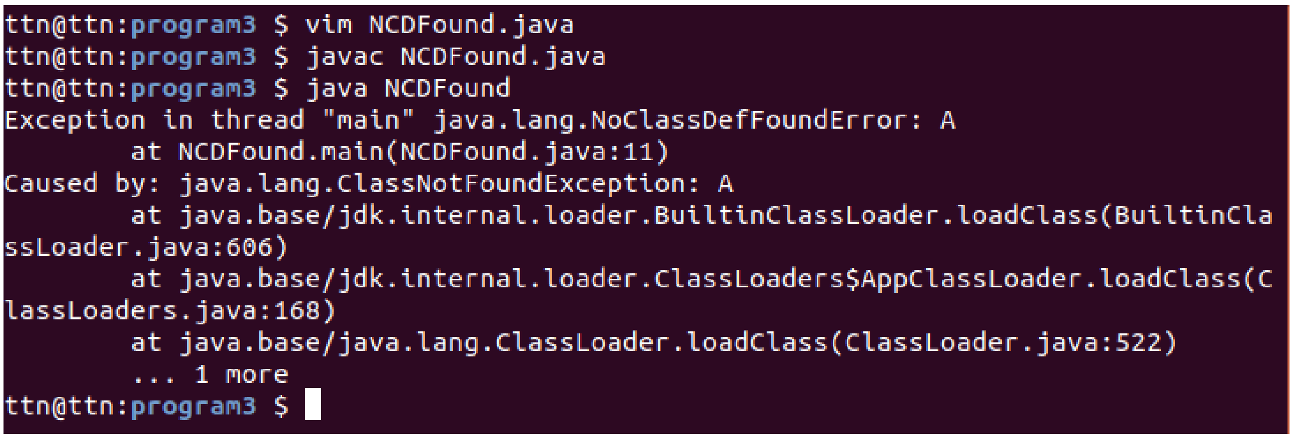
```
public class CNFound {
    public static void main(String args[]) {
        try
        {
            Class.forName("ExampleClass");
        }
        catch (ClassNotFoundException ex)
        {
            ex.printStackTrace();
        }
    }
}
```



```
java.lang.ClassNotFoundException: ExampleClass
        at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinCla
ssLoader.java:606)
        at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(C
lassLoaders.java:168)
        at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:522)
        at java.base/java.lang.Class.forName0(Native Method)
        at java.base/java.lang.Class.forName(Class.java:377)
        at CNFound.main(CNFound.java:6)
```

NoClassDefFoundError – Compile in the program, two .class files will be generated. One is **A.class** and another one is NCDFound**.class**. If you remove the **A.class** file and run the NCDFound**.class** file, Java Runtime System will throw NoClassDefFoundError

```
class A
{
  // some code
}
public class NCDFound
{
    public static void main(String[] args)
    {
        A a = new A();
        System.out.println("hello");
    }
}
```

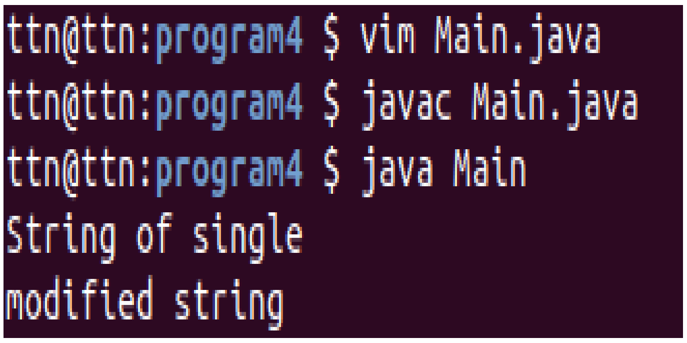


```
ttn@ttn:program3 $ vim NCDFound.java
ttn@ttn:program3 $ javac NCDFound.java
ttn@ttn:program3 $ java NCDFound
Exception in thread "main" java.lang.NoClassDefFoundError: A
        at NCDFound.main(NCDFound.java:11)
Caused by: java.lang.ClassNotFoundException: A
        at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinCla
ssLoader.java:606)
        at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(C
lassLoaders.java:168)
        at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:522)
        ... 1 more
ttn@ttn:program3 $ 
```

4. WAP to create singleton class.

Sol – Program Files Folder name – program4

```
class single{
    private static single single_instance=null;
    public String str;
    private single(){
        str="String of single";
    }
    public static single getSingle_instance(){
        if(single_instance==null){
            single_instance=new single();
        }
        return single_instance;
    }
}

class Main
{
     public static void main(String[] args) {
        single ob=single.getSingle_instance();
        System.out.println(ob.str);
        ob.str="modified string";
        single ob1=single.getSingle_instance();
        System.out.println(ob1.str);
    }

}
```



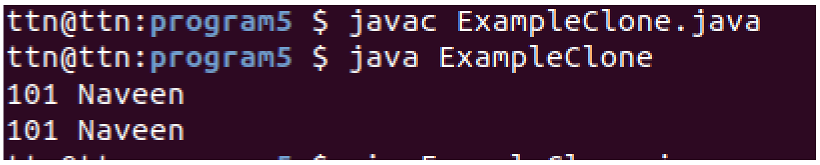
```
ttn@ttn:program4 $ vim Main.java
ttn@ttn:program4 $ javac Main.java
ttn@ttn:program4 $ java Main
String of single
modified string
```

5. WAP to show object cloning in java using cloneable and copy constructor both.

Sol – Program Files Folder name – program5

using cloneable

```
class ExampleClone implements Cloneable{
int rollno;
String name;
ExampleClone(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{
ExampleClone s1=new ExampleClone(101,"Naveen");

ExampleClone s2=(ExampleClone)s1.clone();

System.out.println(s1.rollno+" "+s1.name);
System.out.println(s2.rollno+" "+s2.name);

}catch(CloneNotSupportedException c){}
}
}
```

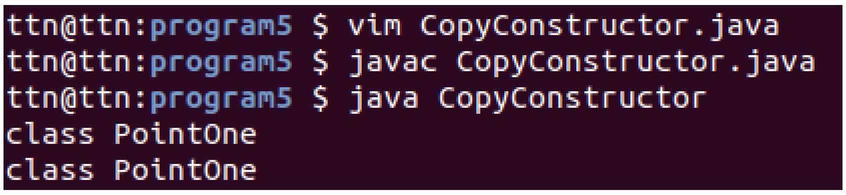


```
ttn@ttn:program5 $ javac ExampleClone.java
ttn@ttn:program5 $ java ExampleClone
101 Naveen
101 Naveen
```

Use Copy Constructor

```
import java.io.*;
class PointOne
{
   private Integer x;
   private Integer y;
   PointOne(int x, int y){
        this.x = x;
```

```
         this.y = y;
    }
    PointOne(PointOne point){
        this.x = point.x;
        this.y = point.y;
    }
}

class PointTwo extends PointOne
{
    private Integer z;

    PointTwo(int x, int y, int z){
        super(x, y);
        this.z = z;
    }



    PointTwo(PointTwo point){
        super(point);
        this.z = point.z;
    }
}
class CopyConstructor
{
    public static void main(String[] args)
    {
        PointOne one = new PointOne(1,2);
        PointTwo two = new PointTwo(1,2,3);

        PointOne clone1 = new PointOne(one);
        PointOne clone2 = new PointOne(two);


        System.out.println(clone1.getClass());

        System.out.println(clone2.getClass());

    }

}
```

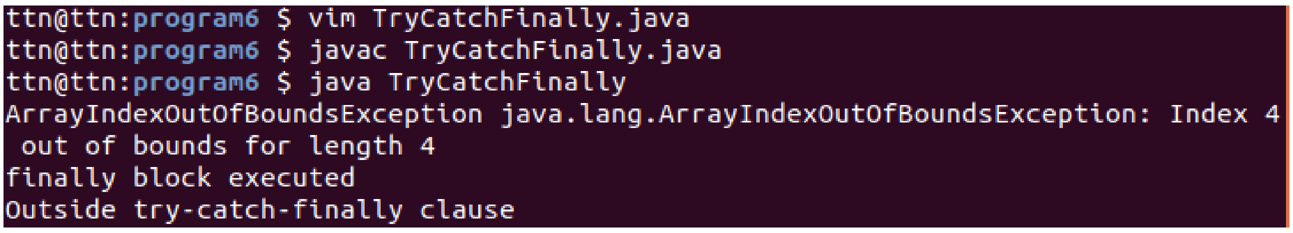


```
ttn@ttn:program5 $ vim CopyConstructor.java
ttn@ttn:program5 $ javac CopyConstructor.java
ttn@ttn:program5 $ java CopyConstructor
class PointOne
class PointOne
```

6. WAP showing try, multi-catch and finally blocks.

Sol – Program Files Folder name – program6

```
class TryCatchFinally

{

    public static void main (String[] args)

    {

        int[] arr = new int[4];


        try

        {

            int i = arr[4];

            System.out.println("Inside try block");

        }


        catch(ArrayIndexOutOfBoundsException ex)

        {

            System.out.println("ArrayIndexOutOfBoundsException "+ ex);

        }

        catch (ArithmeticException ex)

        {
            System.out.println("Arithmetic " + ex);
        }
        catch (NumberFormatException ex)
        {
            System.out.println("Number Format Exception " + ex);
        }
        finally
        {
            System.out.println("finally block executed");
        }
        System.out.println("Outside try-catch-finally clause");
    }
}
```

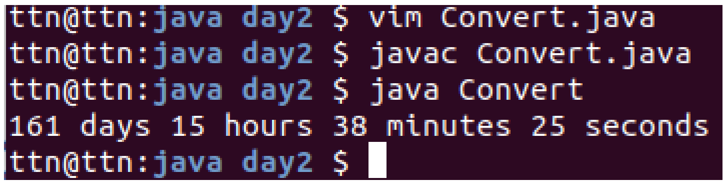


```
ttn@ttn:program6 $ vim TryCatchFinally.java
ttn@ttn:program6 $ javac TryCatchFinally.java
ttn@ttn:program6 $ java TryCatchFinally
ArrayIndexOutOfBoundsException java.lang.ArrayIndexOutOfBoundsException: Index 4
 out of bounds for length 4
finally block executed
Outside try-catch-finally clause
```

7. WAP to convert seconds into days, hours, minutes and seconds.

Sol – Program Files Folder name – program7

```
import java.lang.*;
class Convert
{
    public static void main (String[] args)
    {
        int n = 13966705;
        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 ");

    }

}
```



```
ttn@ttn:java day2 $ vim Convert.java
ttn@ttn:java day2 $ javac Convert.java
ttn@ttn:java day2 $ java Convert
161 days 15 hours 38 minutes 25 seconds
ttn@ttn:java day2 $
```

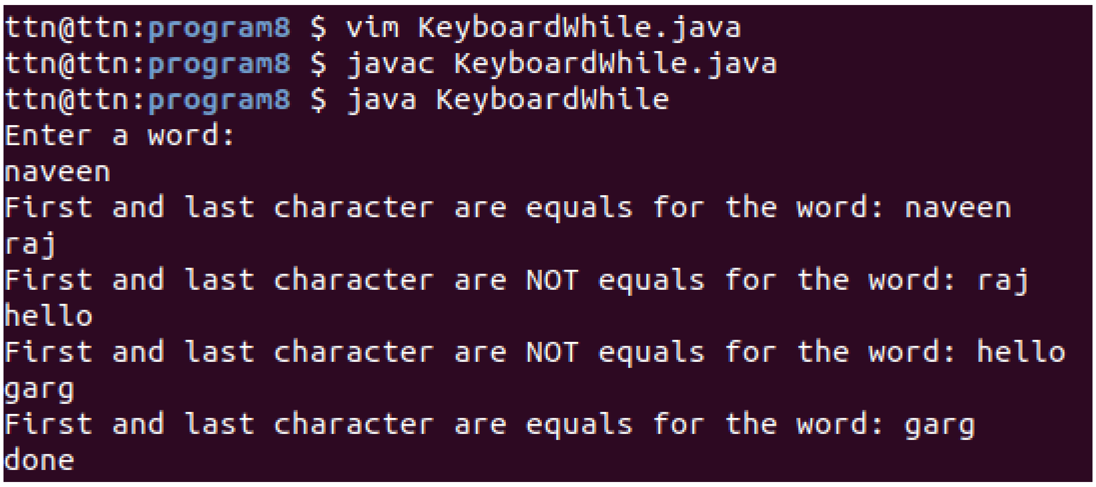
8. WAP to read words from the keyboard until the word done is entered. For each word except done, report whether its first character is equal to its last character. For the required loop, use a
a)while statement
b)do-while statement

Sol – Program Files Folder name – program8

a)while statement

```
import java.util.Scanner;

public class KeyboardWhile

{

        public static void main(String arg[])

        {

                Scanner keyboard = new Scanner(System.in);

                System.out.println("Enter a word:");

                String word = keyboard.next();

                while(!word.equals("done"))

                {

                        if(word.charAt(0) == word.charAt(word.length() - 1))

                        {
```

```
                System.out.println("First and last character are equals for the word: " + word);
            }
            else
            {
                System.out.println("First and last character are NOT equals for the word: " + word);
            }
            word = keyboard.next();
        }
    }
}
```

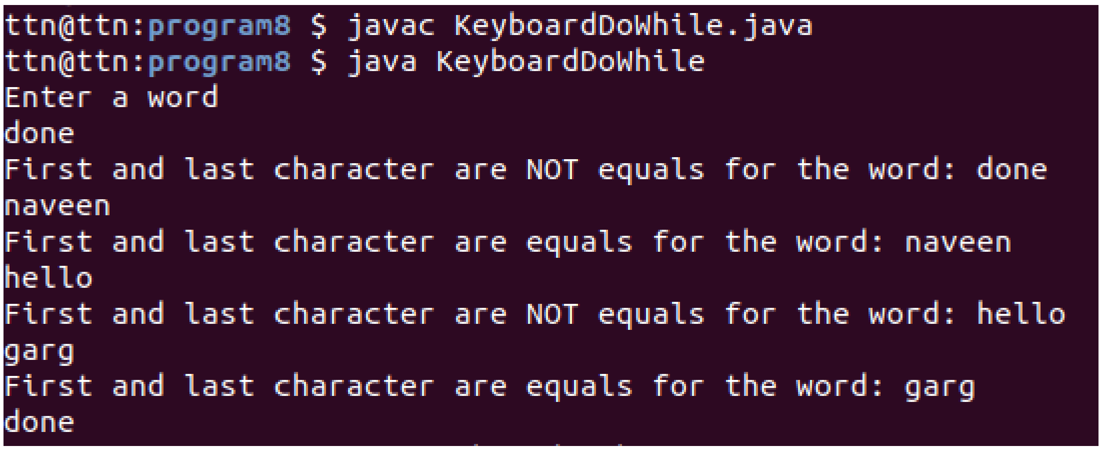


```
ttn@ttn:program8 $ vim KeyboardWhile.java
ttn@ttn:program8 $ javac KeyboardWhile.java
ttn@ttn:program8 $ java KeyboardWhile
Enter a word:
naveen
First and last character are equals for the word: naveen
raj
First and last character are NOT equals for the word: raj
hello
First and last character are NOT equals for the word: hello
garg
First and last character are equals for the word: garg
done
```

b)do-while statement

```
import java.util.Scanner;
public class KeyboardDoWhile
{
    public static void main(String arg[])
    {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter a word");
        String word = keyboard.next();
        do
```

```
		{
			if(word.charAt(0) == word.charAt(word.length() - 1))
			{
				System.out.println("First and last character are equals for
the word: " + word);
			}
			else
			{
				System.out.println("First and last character are NOT equals
for the word: " + word);
			}
			word = keyboard.next();
		}while(!word.equals("done"));
	}
}
```



```
ttn@ttn:program8 $ javac KeyboardDoWhile.java
ttn@ttn:program8 $ java KeyboardDoWhile
Enter a word
done
First and last character are NOT equals for the word: done
naveen
First and last character are equals for the word: naveen
hello
First and last character are NOT equals for the word: hello
garg
First and last character are equals for the word: garg
done
```

9. Design classes having attributes for furniture where there are wooden chairs and tables, metal chairs and tables. There are stress and fire tests for each products.

Sol –

```
public interface Furniture {
	public boolean stressTest();
	public boolean fireTest();
}
```

```
public class FurnitureItem implements Furniture {
        double weight;
        int dimensions;
        public boolean fireTest(){
                //general tests for the furniture
        }
        public boolean stressTest(){
                //general tests for the furniture
        }
}

public class Chair extends FurnitureItem {
        boolean headrest;
        boolean armrest;
        public String toString()
        {
                //say print chair
        }
        public boolean fireTest(){
        //general tests for the chair
        }
        public boolean stressTest(){
        //general tests for the chair
        }
}

public class Table extends FurnitureItem {
        int legs;
        boolean foldable;
        public String toString()
        {
                //say print Table
        }
        public boolean fireTest(){
                //general tests for the Table
        }
        public boolean stressTest(){
                //general tests for the Table
        }
}

public class MetalChair extends Chair {
        boolean painted = true;
        String metaltype="";


        public boolean stressTest() {
                //specific to metal chairs
        }

        public boolean fireTest() {


                //specific to metal chairs
                return  //if everthing is true;
        }
        public String toString()
        {
                //say print
        }


}

public class MetalTable extends Table {
        boolean painted = true;
        String metaltype="";
        public boolean stressTest() {
                //specific to metal table
        }
        public boolean fireTest() {
                //specific to metal table
                return //if everthing is true;
        }
        public String toString()
        {
                //say print
        }
}

public class WoodenTable extends Table {
        boolean polished = true;
        String woodType="";
        public boolean stressTest() {
                //specific to metal table
        }

        public boolean fireTest() {
                //specific to metal table
                return //if everthing is true;
        }
        public String toString()
        {
                //say print
        }
}
public class WoodenChair extends Chair {
        boolean polished = true;
        String woodType="";
        public boolean stressTest() {
                //specific to metal table
        }
        public boolean fireTest() {
                //specific to metal table
                return  //if everthing is true;
        }
        public String toString()
        {
                //say print
        }
}
public class MainClassFurniture {
        public static void main(String[] args){
                //Main class
        }
}
```

10. Design classes having attributes and method(only skeleton) for a coffee shop. There are three different actors in our scenario and i have listed the different actions they do also below

*** Customer**
- Pays the cash to the cashier and places his order, get a token number back
- Waits for the intimation that order for his token is ready
- Upon intimation/notification he collects the coffee and enjoys his drink
(Assumption: Customer waits till the coffee is done, he wont timeout and cancel the order. Customer always likes the drink served. Exceptions like he not liking his coffee, he getting wrong coffee are not considered to keep the design simple.)

*** Cashier**
- Takes an order and payment from the customer
- Upon payment, creates an order and places it into the order queue
- Intimates the customer that he has to wait for his token and gives him his token
(Assumption: Token returned to the customer is the order id. Order queue is unlimited. With a simple modification, we can design for a limited queue size)

*** Barista**
- Gets the next order from the queue
- Prepares the coffee

- Places the coffee in the completed order queue
- Places a notification that order for token is ready

Sol –

```
class Customer{
        int paid;
        int tokenNo;
        int orderId;
        Cashier cashier;
        void placeOrder(int money){
                //code
        }
        void waitForOrder(){
                //code
        }
        void collectOrder(){
                //code
        }
}

class Cashier{
        int amount;
        Barista barista;
        Customer customer;
        int takeOrder(int price,String order){
                return change;
        }
        Queue<Integer> createOrder(int payment,int orderId){
                return que;
        }
        boolean intimate(){
                return coffeeStatus;
        }
}

class Barista{
        Queue<Integer> orderQueue;
```

```
	Queue<Integer> completedQueue;
	int currentOrder;
	boolean coffeeStatus;
	Cashier cashier;

	int nextOrder(){
		return orderId;
	}
	void preparingCoffee(){
		//code
	}
	void updateCompleted(int ){
		//code
	}
	boolean notifyOrder(){
		coffeeStatus;
	}
}
public class MainClass
{
	public static void main(String args[])
	{
		//code
	}
}
```

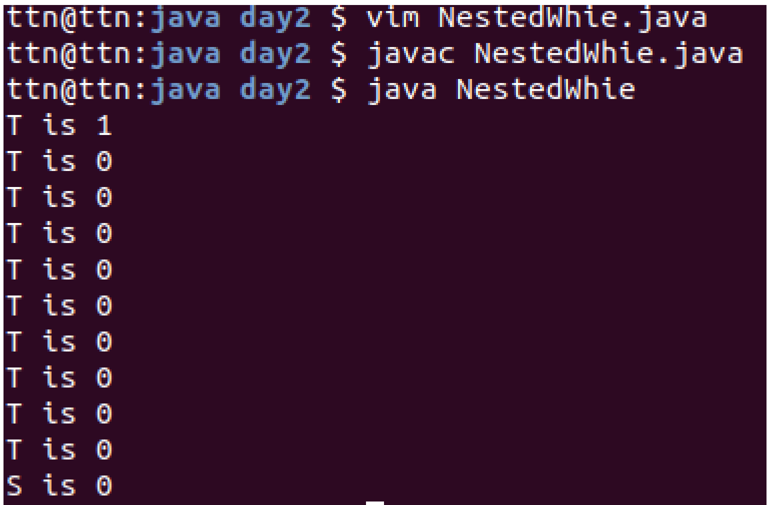
11. Convert the following code so that it uses nested while statements instead of for statements:

```
   int s = 0;
   int t = 1;
   for (int i = 0; i < 10; i++)
   {
   s = s + i;
   for (int j = i; j > 0; j−−)
   {
   t = t * (j - i);
   }
   s = s * t;
   System.out.println("T is " + t);
```

```
    }
    System.out.println("S is " + s);
```

Sol – Program Files Folder name – program11

```
public class NestedWhie
{
        public static void main(String arg[])
        {
                int s = 0;
                int t = 1;
                int i = 0;
                while(i < 10)
                {
                        s = s + i;
                        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);
        }
}
```



```
ttn@ttn:java day2 $ vim NestedWhie.java
ttn@ttn:java day2 $ javac NestedWhie.java
ttn@ttn:java day2 $ java NestedWhie
T is 1
T is 0
T is 0
T is 0
T is 0
T is 0
T is 0
T is 0
T is 0
T is 0
S is 0
```

12.What will be the output on new Child(); ?

```
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");
    }
}
```

Output:

```
static – grandparent

static – parent
```

static - child

instance – grandparent

constructor – grandparent

instance - parent

constructor – parent

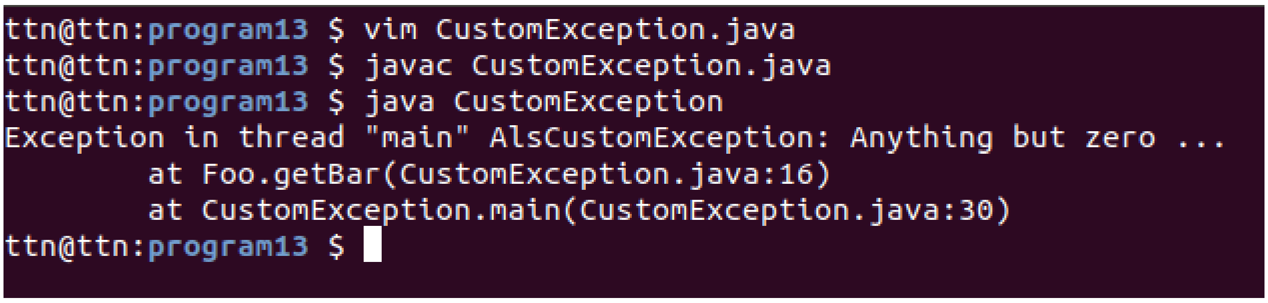
instance – child

constructor - child

Q13. Create a custom exception that do not have any stack trace.

Sol – Program Files Folder name – program13

```
class AlsCustomException extends RuntimeException
{
  public AlsCustomException(String message)
  {
    super(message);
  }
}

class Foo
{
  public String getBar(int i)
  throws AlsCustomException
  {
    if (i == 0)
    {
      throw new AlsCustomException("Anything but zero ...");
    }
    else
    {
      return "Thanks";
    }
  }
}

public class CustomException
{
  public static void main(String[] args)
  {
    Foo foo = new Foo();
    String bar = foo.getBar(0);
  }
}
```



```
ttn@ttn:program13 $ vim CustomException.java
ttn@ttn:program13 $ javac CustomException.java
ttn@ttn:program13 $ java CustomException
Exception in thread "main" AlsCustomException: Anything but zero ...
        at Foo.getBar(CustomException.java:16)
        at CustomException.main(CustomException.java:30)
ttn@ttn:program13 $ 
```