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 1.

```java
package problemDay2;

import java.util.Scanner;

public class Ques1User extends Ques1book {

    String uname;
    String stream;
    String rollno;

    void getUDetails()
    {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter details of user");
        System.out.println("Enter user name: ");
        uname=input.next();
        System.out.println("Enter user stream: ");
        stream=input.next();
        System.out.println("Enter roll number: ");
        rollno=input.next();
    }

    void showUDetails()
    {
        System.out.println("The book issued to "+uname+" of "+stream+" and having roll number "+rollno);
    }

    public static void main(String[] args) {
        Ques1User ob=new Ques1User();
        ob.libraryDetails();
        ob.getDetails();
```

Ques1User › getUDetails()

```java
        Scanner input = new Scanner(System.in);
        System.out.println("Enter details of user");
        System.out.println("Enter user name: ");
        uname=input.next();
        System.out.println("Enter user stream: ");
        stream=input.next();
        System.out.println("Enter roll number: ");
        rollno=input.next();
    }

    void showUDetails()
    {
        System.out.println("The book issued to "+uname+" of "+stream+" and having roll number "+rollno);
    }

    public static void main(String[] args) {
        Ques1User ob=new Ques1User();
        ob.libraryDetails();
        ob.getDetails();
        ob.showDetails();
        ob.getUDetails();
        ob.showUDetails();
    }

}
```

```java
package problemDay2;

import java.util.Scanner;

public class Ques1book extends Ques1 {
    String bookname;
    String writer;
    String bookno;

    void getDetails()
    {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter details of book");
        System.out.println("Enter book name: ");
        bookname=input.next();

        System.out.println("Enter book author:");
        writer=input.next();

        System.out.println("Enter book number: ");
        bookno=input.next();
    }

    void showDetails()
    {
        System.out.println("The book issued is "+bookname+" written by "+writer+" and having book number "+bookno);
    }
}
```

Ques1book

```java
package problemDay2;

import java.util.Scanner;

public class Ques1 {

    String lname="Central Library";
    String librarian="M.k.Gupta";
    void libraryDetails()
    {
        System.out.println(""+lname);
        System.out.println(""+librarian);
    }

}
```

Ques1User

```
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Central Library
M.k.Gupta
Enter details of book
Enter book number:
1233
Enter book author:
M.K.Das
Enter book name:
The Dreams
The book issued is The written by M.K.Das and having book number 1233
Enter details of user
Enter user name:
Aman
Enter user stream:
CSE
Enter roll number:
21
The book issued to Aman of CSE and having roll number 21

Process finished with exit code 0
```

2. WAP to sorting string without using string Methods?.

Sol 2.

```java
package problemDay2;

public class Ques2 {
    public static void main(String[] args)
    {
        String str="sortthisstring";

        char temp = 0;

        char arr[] = str.toCharArray();
        for (int i = 0; i < arr.length; i++)
        {
            for (int j = 0; j < arr.length; j++)
            {
                if (arr[j] > arr[i])
                {
                    temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
        }
        System.out.println("Original: "+str);
        for (int i = 0; i < arr.length; i++)
        {
            System.out.print(arr[i]);
        }
    }
}
```

Ques2 › main()

```
Ques2
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Original: sortthisstring
ghiinorrsssttt
Process finished with exit code 0
```

3. WAP to produce NoClassDefFoundError and ClassNotFoundException exception.

Sol 3.

```java
package problemDay2;

public class Ques3 {

    public static void main(String args[]) {
        try
        {
            Class.forName("ClassAman");
        }
        catch (ClassNotFoundException ex)
        {
            System.out.println(ex);
            //ex.printStackTrace();
        }

        try {
            // The following line would throw ExceptionInInitializerError
            Ques3Cal calculator1 = new Ques3Cal();
        }
        catch (Throwable t) {
            System.out.println(t);
        }
        Ques3Cal calculator2 = new Ques3Cal();
    }
}
```

Ques3 › main()

```java
package problemDay2;

public class Ques3Cal {
    static int undefined = 1 / 0;
}
```

Ques3

```
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
java.lang.ClassNotFoundException: ClassAman
java.lang.ExceptionInInitializerError
Exception in thread "main" java.lang.NoClassDefFoundError: Could not initialize class problemDay2.Ques3Cal
    at problemDay2.Ques3.main(Ques3.java:23)

Process finished with exit code 1
```

4. WAP to create singleton class.

Sol 4.

```java
}
public class Ques4 {
    public static void main(String args[])
    {
        // instantiating Singleton class with variable x
        Singleton x = Singleton.getInstance();

        // instantiating Singleton class with variable y
        Singleton y = Singleton.getInstance();

        // instantiating Singleton class with variable z
        Singleton z = Singleton.getInstance();

        // changing variable of instance x
        x.s = (x.s).toUpperCase();

        System.out.println("String from x is " + x.s);
        System.out.println("String from y is " + y.s);
        System.out.println("String from z is " + z.s);
        System.out.println("\n");

        // changing variable of instance z
        z.s = (z.s).toLowerCase();

        System.out.println("String from x is " + x.s);
        System.out.println("String from y is " + y.s);
        System.out.println("String from z is " + z.s);
    }

}
```

Singleton › getInstance()

```
Ques4
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
String from x is HELLO I AM A STRING PART OF SINGLETON CLASS
String from y is HELLO I AM A STRING PART OF SINGLETON CLASS
String from z is HELLO I AM A STRING PART OF SINGLETON CLASS


String from x is hello i am a string part of singleton class
String from y is hello i am a string part of singleton class
String from z is hello i am a string part of singleton class

Process finished with exit code 0
```

inal ▶ 4: Run 5: Debug 6: TODO

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

Sol 5.

```java
package problemDay2;

class Test
{
    int x, y;
    Test()
    {
        x = 10;
        y = 20;
    }
}
public class Ques5 {
    public static void main(String[] args)
    {
        Test ob1 = new Test();

        System.out.println(ob1.x + " " + ob1.y);

        // Creating a new reference variable ob2
        // pointing to same address as ob1
        Test ob2 = ob1;

        // Any change made in ob2 will be reflected
        // in ob1
        ob2.x = 100;

        System.out.println(ob1.x+" "+ob1.y);
        System.out.println(ob2.x+" "+ob2.y);
    }
}
```

Ques5

```
Ques5
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
10 20
100 20
100 20

Process finished with exit code 0
```

```java
package problemDay2;

class Test1
{
    int x, y;
}
class Test2 implements Cloneable
{
    int a;
    int b;
    Test1 c = new Test1();
    public Object clone() throws
            CloneNotSupportedException
    {
        return super.clone();
    }
}

public class Ques5Clone {
    public static void main(String args[]) throws
            CloneNotSupportedException
    {
        Test2 t1 = new Test2();
        t1.a = 10;
        t1.b = 20;
        t1.c.x = 30;
        t1.c.y = 40;

        Test2 t2 = (Test2)t1.clone();

        // Creating a copy of object t1 and passing
```

Ques5Clone

```java
            CloneNotSupportedException
    {
        Test2 t1 = new Test2();
        t1.a = 10;
        t1.b = 20;
        t1.c.x = 30;
        t1.c.y = 40;

        Test2 t2 = (Test2)t1.clone();

        // Creating a copy of object t1 and passing
        //  it to t2
        t2.a = 100;

        // Change in primitive type of t2 will not
        // be reflected in t1 field
        t2.c.x = 300;

        // Change in object type field will be
        // reflected in both t2 and t1(shallow copy)
        System.out.println(t1.a + " " + t1.b + " " +
                t1.c.x + " " + t1.c.y);
        System.out.println(t2.a + " " + t2.b + " " +
                t2.c.x + " " + t2.c.y);
    }
}
```

Ques5Clone

```
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
10 20 300 40
100 20 300 40

Process finished with exit code 0
```

minal ▶ 4: Run 5: Debug 6: TODO

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

Sol 6.

```java
public class Ques6 {
    public static void main(String[] args) {


        try
        {
          int x=1;
          int y=0;
          int z;
          z= x/y;
        }

        catch(ArrayIndexOutOfBoundsException e)
        {
            System.out.println(e);
        }

        catch(ArithmeticException e)
        {
            System.out.println(e);
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
        finally
        {
            System.out.println("reached finally block");
        }
    }
}
```

Ques6 › main()

```
Ques6
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
java.lang.ArithmeticException: / by zero
reached finally block

Process finished with exit code 0
```

inal ▶ 4: Run 5: Debug 6: TODO

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

Sol 7.

```java
package problemDay2;

import java.util.Scanner;

public class Ques7 {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int secs,tot;
        int rem=0;
        System.out.println("Enter time in seconds:");
        secs= input.nextInt();
        tot=secs;
        int days= (secs/86400);
        rem= secs-(days*86400);
        int hour= (rem/3600);
        rem=rem-(hour*3600);
        int min= (rem/60);
        rem=rem-(min*60);
        System.out.println("Total time in seconds: "+tot);
        System.out.println("Time in days "+days+" hours "+hour+" minutes "+min+" seconds "+rem);
    }
}
```

Ques7 › main()

```
Ques7
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Enter time in seconds:
109288321
Total time in seconds: 109288321
Time in days 1264 hours 21 minutes 52 seconds 1

Process finished with exit code 0
```

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 8.

```java
import java.util.Scanner;

public class Ques8 {
    public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter a word");
        String word = keyboard.next();
        //while-loop

        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();
        }

        //do-while loop

        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")); }
}
```

Ques8 › main()

```
Ques8
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Enter a word
aman
First and last character are NOT equals for the word: aman
is
First and last character are NOT equals for the word: is
my
First and last character are NOT equals for the word: my
name
First and last character are NOT equals for the word: name
done
First and last character are NOT equals for the word: done
this
First and last character are NOT equals for the word: this
is
First and last character are NOT equals for the word: is
while
First and last character are NOT equals for the word: while
done

Process finished with exit code 0
```

inal 0: Messages 4: Run 5: Debug 6: TODO

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 9.

```java
package problemDay2;

import java.util.Scanner;

public class Ques9 {

    String ftest;
    String stest;
    Scanner input = new Scanner(System.in);
    void fireTest()
    {
        System.out.println("Does it passed fire test?");
        ftest=input.next();
        //System.out.println(""+ftest);
    }

    void stressTest()
    {
        System.out.println("Does it passed stress test?");
        stest=input.next();
        //System.out.println(""+stest);
    }
}
```

```java
package problemDay2;
import java.util.Scanner;

public class Ques9Chair extends Ques9{
    String type;
    int cost;
    Scanner input = new Scanner(System.in);
    void prop(){
        System.out.println("Type of Chair(metal or wood)?");
        type=input.next();
        System.out.println("Cost of Chair:");
        cost=input.nextInt();
    }
    public static void main(String[] args) {

        Ques9Chair obj=new Ques9Chair();
        obj.prop();
        obj.fireTest();
        obj.stressTest();
        System.out.println("It is a "+obj.type+" table having cost "+obj.cost);
        System.out.println("Result of fire test: "+obj.ftest);
        System.out.println("Result of stress test: "+obj.stest);
    }

}
```

```
Ques9Chair
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Type of Chair(metal or wood)?
wood
Cost of Chair:
4000
Does it passed fire test?
no
Does it passed stress test?
yes
It is a wood table having cost 4000
Result of fire test: no
Result of stress test: yes

Process finished with exit code 0
```

```java
package problemDay2;
import java.util.Scanner;

public class Ques9Table extends Ques9{

    String type;
    int cost;
    Scanner input = new Scanner(System.in);
    void prop(){
    System.out.println("Type of Table(metal or wood)?");
    type=input.next();
    System.out.println("Cost of Table:");
    cost=input.nextInt();
    }
    public static void main(String[] args) {

        Ques9Table obj=new Ques9Table();
        obj.prop();
        obj.fireTest();
        obj.stressTest();
        System.out.println("It is a "+obj.type+" table having cost "+obj.cost);
        System.out.println("Result of fire test: "+obj.ftest);
        System.out.println("Result of stress test: "+obj.stest);
    }
}
```

```
Ques9Table
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Type of Table(metal or wood)?
metal
Cost of Table:
10000
Does it passed fire test?
yes
Does it passed stress test?
yes
It is a metal table having cost 10000
Result of fire test: yes
Result of stress test: yes

Process finished with exit code 0
```

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 10.

This only displays the order in which methods will be executed.

```java
package problemDay2;

import java.util.Scanner;

class Customer{
    String name;
    String oname;
    static String ispaid;
    int orderno;
    Scanner input = new Scanner(System.in);
    void getDetails(){
        System.out.println("1. Taking Customer details");
       /* String name = input.next();
        System.out.println("Customer order name:");
        String oname = input.next();
        System.out.println("Has Customer paid?:(y/n)");
        String ispaid = input.next();*/
        ispaid="y";
    }
}
```

```java
class Cashier extends Customer{
    int money;
    String corder;
    int corderno;
    int i=1;
    void setDetails(){
        System.out.println("2. Forwarding Customer order details if money is paid");
        if(Customer.ispaid=="y" || Customer.ispaid=="Y")
        {
            //corderno=i;
            //i++;
        }
        else
        {
            System.out.println("Please ask customer to pay.");
        }

    }
}
```

```java
public class Barista extends Cashier {
    String orname;
    String status;
    String ornum;
    String feedback;

    void showDetails()
    {
        System.out.println("3. Customer order details received \n - Preparing order");
        System.out.println("4. Taking order feedback");
    }
    public static void main(String[] args) {

        Barista b= new Barista();
        b.getDetails();
        b.setDetails();
        b.showDetails();
    }
}
```

Cashier › money

CDlatch × Barista ×

```
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
1. Taking Customer details
2. Forwarding Customer order details if money is paid
3. Customer order details received
 - Preparing order
4. Taking order feedback

Process finished with exit code 0
```

0: Messages 4: Run 5: Debug 6: TODO

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 11.

```java
package problemDay2;

public class Ques11 {
    public static void main(String[] args) {
        int s = 0;
        int t = 1;
        int i=0;
        int j;
        while(i < 10)
        {
            s = s + i;
            j=i;
            while( j > 0)
            {
                t = t * (j - i);
                j--;
            }
            s = s * t;
            i++;
            System.out.println("T is " + t);
        }
        System.out.println("S is " + s);
    }
}
```

Ques11 › main()

```
Ques11
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
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

Process finished with exit code 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");
    }
}
```

Sol 12.

```
package problemDay2;

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");
    }
}
```

Child

```java
    public Grandparent() {
        System.out.println("constructor - grandparent");
    }
}

public class Child extends Parent {

    public Child() {
        System.out.println("constructor - child");
    }

    static {
        System.out.println("static - child");
    }

    {
        System.out.println("instance - child");
    }

    public static void main(String[] args) {

        Child ob=new Child();
    }
}
```

```
Child
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
static - grandparent
static - parent
static - child
instance - grandparent
constructor - grandparent
instance - parent
constructor - parent
instance - child
constructor - child

Process finished with exit code 0
```

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

Sol 13.

```java
package problemDay2;

class MyCustomException extends Exception
{
    public MyCustomException(String message) { super(message); }
}

class Foo
{ public String getBar(int i) throws MyCustomException
    { if (i == 0)
        {// throw our custom exception
            throw new MyCustomException("Anything can be entered but not zero ...");
        }
        else
        {return "Thanks";
        } }}

public class Ques13 {
    public static void main(String[] args)
    {   // create a new foo
        Foo foo = new Foo();

        try
        {   // intentionally throw our custom exception by
            // calling getBar with a zero
            String bar = foo.getBar( i: 0);
        }
        catch (MyCustomException e)
        {
            // print the stack trace
            e.printStackTrace(); } }}
```

Foo › getBar()

Ques13

```
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
problemDay2.MyCustomException: Anything can be entered but not zero ...
    at problemDay2.Foo.getBar(Ques13.java:15)
    at problemDay2.Ques13.main(Ques13.java:29)

Process finished with exit code 0
```