**Time allowed: 90 Minutes Max. Marks: 40**

**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

1) Which of the following class definitions defines a legal abstract class?

a) class A { abstract void unfinished() { } }

b) class A { abstract void unfinished(); }

**c) abstract class A { abstract void unfinished(); }**

d) public class abstract A { abstract void unfinished(); }

2) Which of these keywords is used to define packages in Java?

a) pkg

b) Pkg

**c) package**

d) Package

3) Package in Java is a mechanism to encapsulate a \_\_\_\_\_\_\_\_\_\_\_\_\_\_.

a) Classes

b) Interfaces

c) Sub Packages

**d) All of the above**

4) Which of the following statements regarding abstract classes are true?

a) An abstract class can be extended.

b) A subclass of a non-abstract superclass can be abstract.

c) A subclass can override a concrete method in a superclass to declare it abstract.

**d) All of the above**

5) Which of these method of class String is used to obtain length of String object?

a) get()

b) Sizeof()

c) lengthof()

**d) length()**

6) Multithreading is also called as \_\_\_\_\_\_\_\_\_\_\_\_

**a) Concurrency**

b) Simultaneity

c) Crosscurrent

d) Recurrent

7) Which exception is thrown when divide by zero statement executes?

a) NumberFormatException

**b) ArithmeticException**

c) NullPointerException

d) None of these

8) Which keyword is used to explicitly throw an exception?

a) try

b) throwing

c) catch

**d) throw**

9) Which exception is thrown when divide by zero statement executes?

a) NumberFormatException

**b) ArithmeticException**

c) NullPointerException

d) None of these

10) In Java, declaring a class abstract is useful

a) To prevent developers from further extending the class.

**b) When it doesn't make sense to have objects of that class.**

c) When default implementations of some methods are not desirable.

d) To force developers to extend the class not to use its capabilities.

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

11) Determine output of the following code.

interface A { }

class C { }

class D extends C { }

class B extends D implements A { }

public class Test extends Thread{

public static void main(String[] args){

B b = new B();

if (b instanceof A)

System.out.println("b is an instance of A");

if (b instanceof C)

System.out.println("b is an instance of C");

}

}

a) Nothing.

b) b is an instance of A.

c) b is an instance of C.

**d) b is an instance of A followed by b is an instance of C.**

12) Choose a correct way of importing all the classes in the below java program with packages.

//Cat.java

package animals;

public class Cat { }

//Dog.java

package animals;

public class Dog { }

//PackageTesting.java

//import statements

public class

{

//new Cat();

//new Dog();

}

a) import animals.\*;

b) import animals.cat;

import animals.Dog;

**c) Both A and B**

d) None

13) What will be the output of below statements?

String s1 = "abc";

String s2 = "def";

System.out.println(s1.compareTo(s2));

a) 0

b) true

**c) -3**

d) false

14) What will be the output of the following Java code?

class multithreaded\_programing

{

public static void main(String args[])

{

Thread t = Thread.currentThread();

System.out.println(t);

}

}

a) Thread[5,main]

b) Thread[main,5]

c) Thread[main,0]

**d) Thread[main,5,main]**

15) What is the priority of the thread in the following Java Program?

class multithreaded\_programing

{

public static void main(String args[])

{

Thread t = Thread.currentThread();

System.out.println(t);

}

}

a) 4

**b) 5**

c) 0

d) 1

**SECTION-C(Coding Question) (2x5 marks=5 marks)**

Q16) Given a string str consisting of only two characters 'a' and 'b'. You need to find the minimum steps required to make the string empty by removing consecutive a's and b's.

**Input:** str = "bbaaabb"

**Output:** 2

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | aababaa | Bbaab | Bbaabab |
| **Output** | 3 | 2 | 3 |

Solution :

**import java.util.\*;**

**import java.io.\*;**

**import java.lang.\*;**

**public class MyClass{**

**public static void main(String [] args) throws IOException{**

**Solution obj = new Solution();**

**System.out.println(obj.minSteps("abbaabb"));**

**}**

**}**

**class Solution{**

**int minSteps(String str) {**

**int n=str.length();**

**int ind=0;**

**int consA=0, consB=0;**

**int tempA=0, tempB=0;**

**while(ind<n){**

**while(ind<n && str.charAt(ind)=='a'){**

**ind++; tempA=1;**

**}**

**while(ind<n && str.charAt(ind)=='b'){**

**ind++; tempB=1;**

**}**

**consA+=tempA; consB+=tempB;**

**tempA=0; tempB=0;**

**}**

**return Math.min(consA,consB)+1;**

**}**

**}**

Q17) Given an expression string x. Examine whether the pairs and the orders of {,},(,),[,] are correct in exp.

For example, the function should return 'true' for exp = [()]{}{[()()]()} and 'false' for exp = [(]).

**Input:** {([])}

**Output:** true

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | [{([])}] | ([] | (()} |
| **Output** | true | False | false |

Solution :

**import java.util.\*;**

**import java.io.\*;**

**import java.lang.\*;**

**public class MyClass**

**{**

**public static void main(String args[])**

**{**

**//reading the string**

**String st = "({[]})";**

**if(new Solution().ispar(st) == true)**

**System.out.println("balanced");**

**else**

**System.out.println("not balanced");**

**}**

**}**

**class Solution**

**{**

**//Function to check if brackets are balanced or not.**

**static boolean ispar(String x)**

**{**

**Stack<Character> st = new Stack<>();**

**for(int i=0;i<x.length();i++)**

**{**

**char c = x.charAt(i);**

**if(st.isEmpty())**

**st.push(c);**

**else if(c == ']' && st.peek() == '[' || c == '}' && st.peek() == '{' || c == ')' && st.peek() == '('){**

**st.pop();}**

**else**

**st.push(c);**

**}**

**if(st.isEmpty())**

**return true;**

**return false;**

**}**

**}**

**SECTION-D (Coding Question)(1x10 mark=10 mark)**

Q18) Create a class Student with attributes roll no, name, age and course. Initialize values through parameterized constructor. If age of student is not in between 15 and 21 then generate user-defined exception "AgeNotWithinRangeException". If name contains numbers or special symbols raise exception "NameNotValidException". Define the two exception classes.

**Input:**

Enter roll number: 34

Enter name: @#$$#

Enter age: 1

Enter course: CS

**Output:**

Name is not Valid..Please ReEnter the Name

Age is not between 15 and 21. please ReEnter the Age

roll Name Age Course

---------------------

34 null 0 CS

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | Enter roll number: 32  Enter name: #user  Enter age: 21  Enter course: Mech | Enter roll number: 64  Enter name: User  Enter age: 25  Enter course: IT | Enter roll number: 45  Enter name: Raunak  Enter age: 19  Enter course: Civil |
| **Output** | Name is not Valid..Please ReEnter the Name  roll Name Age Course  ---------------------  32 null 21 Mech | Age is not between 15 and 21. please ReEnter the Age  roll Name Age Course  ---------------------  64 User 0 IT | roll Name Age Course  ---------------------  45 Raunak 19 Civil |

Solution :

**import java.io.\*;**

**class AgeNotWithInRangeException extends Exception**

**{**

**public String toString()**

**{**

**return ("Age is not between 15 and 21. please ReEnter the Age");**

**}**

**}**

**class NameNotValidException extends Exception**

**{**

**public String toString()**

**{**

**return ("Name is not Valid..Please ReEnter the Name");**

**}**

**}**

**class Student**

**{**

**int roll,age;**

**String name,course;**

**Student()**

**{**

**roll=0;**

**name=null;**

**age=0;**

**course=null;**

**}**

**Student(int r,String n,int a,String c)**

**{**

**roll=r;**

**course=c;**

**int l,temp=0;**

**l = n.length();**

**for(int i=0;i<l;i++)**

**{**

**char ch;**

**ch=n.charAt(i);**

**if(ch<'A' || ch>'Z' && ch<'a' || ch>'z')**

**temp=1;**

**}**

**/\*———-Checking Name——————–\*/**

**try**

**{**

**if(temp==1)**

**throw new NameNotValidException();**

**else**

**name=n;**

**}**

**catch(NameNotValidException e2)**

**{**

**System.out.println(e2);**

**}**

**/\*———-Checking Age——————–\*/**

**try**

**{**

**if(a>=15 && a<=21)**

**age=a;**

**else**

**throw new AgeNotWithInRangeException();**

**}**

**catch(AgeNotWithInRangeException e1)**

**{**

**System.out.println(e1);**

**}**

**}**

**void display()**

**{**

**System.out.println("roll Name Age Course");**

**System.out.println("---------------------");**

**System.out.println(roll+" "+name+" "+age+" " +course);**

**}**

**}**

**class JavaProgram**

**{**

**public static void main(String args[])throws IOException**

**{**

**BufferedReader br=new BufferedReader(new InputStreamReader(System.in));**

**System.out.print("Enter roll number: ");**

**int roll = Integer.parseInt(br.readLine());**

**System.out.print("\nEnter name: ");**

**String name = br.readLine();**

**System.out.print("\nEnter age: ");**

**int age = Integer.parseInt(br.readLine());**

**System.out.print("\nEnter course: ");**

**String course = br.readLine();**

**Student s = new Student(roll,name,age,course);**

**s.display();**

**}**

**}**