IRC_SKCT_CSE_IT_AIDS_OOPS_CA1

Test Summary

- No. of Sections: 2
- No. of Questions: 25
- Total Duration: 110 min

Section 1 - MCQ

Section Summary

- No. of Questions: 20
- Duration: 20 min

Additional Instructions:

None

Q1. Output of the below code snippet

```
1  class Main
2  {
3      public static void main(String args[])
4      {
5          Object obj = new Object();
6          System.out.print(obj.getClass());
7      }
8  }
9
```

- Object
- class Object
- class java.lang.Object
- Compilation Error

Q2. A single class can have

- only one instance
- two instances
- any number of instances
- None of the mentioned options

Q3. Which of these operators is used to allocate memory for an object?

- malloc
- alloc

new

give

Q4. Output of the below code snippet

```
1  class Main
2  {
3     public static void main(String args[])
4     {
5        Object obj = new Object();
6        System.out.print(obj.getclass());
7     }
8  }
9
```

Object

class Object

class java.lang.Object

Compilation Error

Q5. Which statement about a no-argument constructor is true?

The Java compiler will always insert a default no-argument constructor if you do not define a no-argument constructor in your class.

In order for a class to call super() in one of its constructors, its parent class must explicitly implement a no-argument constructor.

If a class extends another class that has only one constructor that takes a value, then the child class must explicitly declare at least one constructor.

A class may contain more than one no-argument constructor.

Q6. What is not the use of “this” keyword in Java?

Passing itself to another method

Calling another constructor in constructor chaining

Referring to the instance variable when local variable has the same name

Passing itself to method of the same class

Q7. Which of the following is not a reason to use encapsulation when designing a class?

Promote usability by other developers.

- Maintain class data integrity of data elements.
- Prevent users from modifying the internal attributes of a class.
- Increase concurrency and improve performance.

Q8. What is the process of defining a method in terms of itself, that is a method that calls itself?

- Recursion
- Polymorphism
- Abstraction
- Encapsulation

Q9. Predict the output of the following program?

```
1  abstract class demo
2  {
3     public int a;
4     demo()
5     {
6        a = 10;
7     }
8
9     abstract public void set();
10
11    abstract final public void get();
12
13 }
14
15 class Test extends demo
16 {
17
18    public void set(int a)
19    {
20       this.a = a;
21    }
22
23    final public void get()
24    {
25       System.out.println("a = " + a);
26    }
27
28    public static void main(String[] args)
29    {
30       Test obj = new Test();
31       obj.set(20);
32       obj.get();
33    }
34 }
35
36
```

- a = 10
- a = 20
- Compilation error
- Run time error

Q10. Can abstract classes be used in multilevel inheritance?

- Yes, always
- Yes, only one abstract class
- No, abstract class doesn’t have constructors
- No, never

Q11. Suppose the abstract class Message is defined below

A concrete subclass of Message, FrenchMessage, is defined. Which methods must FrenchMessage define?

```
1  public abstract class Message
2  {
3     private String value;
4     public Message(String initial)
5     {
6        value = initial;
7     }
8     public String getMessage()
9     {
10       return value;
11    }
12    public abstract String translate();
13 }
14
```

- translate() only
- getMessage() only
- The FrenchMessage constructor and translate() only
- The FrenchMessage constructor, getMessage(), and translate()

Q12. What is the output of the below Java program?

```
1  public class TestingConstructor
2  {
3    void TestingConstructor()
4    {
5      System.out.println("Amsterdam");
6    }
7
8    TestingConstructor()
9    {
10     System.out.println("Antarctica");
11   }
12
13   public static void main(String[] args)
14   {
15     TestingConstructor tc = new TestingConstructor();
16   }
17 }
```

- Antarctica
- Amsterdam
- No output

Compiler error

Q13. Which of the following can be overloaded?

Methods

Constructors

Both the mentioned

None of the mentioned

Q14. Find the compilation error in the below code

```
1  public class Test
2  {
3    Test(){}  //line A
4
5    static void Test(){ this(); }   //line B
6
7    public static void main(String args[])   //line C
8    {
9      Test();  //line D
10   }
11 }
```

At line A, constructor Tester must be marked public like its class

At line B, constructor call

At line C, compilation error, ambiguity problem, compiler can't determine whether a constructor

At Line D

Q15. What will be the output of the below code?

```
1  class Box
2  {
3     int width;
4     int height;
5     int length;
6     int volume;
7
8     Box()
9     {
10       width = 5;
11       height = 5;
12       length = 6;
13    }
14    void volume()
15    {
16       volume = width*height*length;
17    }
18 }
19 class Constructor_output
20 {
21    public static void main(String args[])
22    {
23       Box obj = new Box();
24       obj.volume();
25       System.out.println(obj.volume);
26    }
27 }
```

100

150

170

200

Q16. What would be the behaviour if one parameterized constructor is explicitly defined?

Compilation error

Compilation succeeds

Runtime error

Compilation succeeds but at the time of creating object using default constructor, it throws compilation error

Q17. Which correctly fills in the blank to print 2017-01-15?

I. f.format(hatDay)
II. f.formatDate(hatDay)
III. hatDay.format(f)

```
1  import java.util.Date;
2  import java.util.Locale;
3  import java.text.DateFormat;
4  import java.time.*;
5  import java.time.format.DateTimeFormatter;
6  public class Main
7  {
8     public static void main(String[] args)
9     {
10
11        LocalDate date = LocalDate.now();
12        LocalDate hatDay = LocalDate.of(2020, Month.JANUARY, 15);
13        DateTimeFormatter f = DateTimeFormatter.ISO_DATE;
14        System.out.println(_____);
15    }
16 }
17
```

I

III

I and III

II and III

Q18. Which of these represents the earliest date/ time?

2017-02-15T03: 00 + 01: 00[Europe/ Berlin]

2017-02-15T04: 00 + 02: 00[Europe/ Helsinki]

2017-02-15T05: 00 + 01: 00[Europe/ Warsaw]

None of the above. We have a tie.

Q19. Which class has a getSeconds() method?

Only the Duration class

Only the Period

Both the Duration and Period classes

Neither class

Q20. Which one of the following classes is best suited for storing timestamp values of application events in a file?

java.time.ZoneId class

java.time.ZoneOffset class

java.time.Instant class

java.time.Duration class

Section 2 - CODING

Section Summary

- No. of Questions: 5
- Duration: 90 min

Additional Instructions:

None

Q1. Write two subclasses named Dog and Cat to override the method Animal.

Input Format

No console input.

Output Format

Print the String from subclass named Dog and Cat in seperate lines.

Sample Input

Sample Output

```
Dog
Cat
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q2. Write a Multiply function for two integers and use overload the function by changing the parameter for double data type.

Input Format

Input two Integers in a separate line & two Double in a separate line.

Output Format

One Integer value and Double value after performing multiplication in a separate line.

Sample Input

```
2
3
1.2
2.3
```

Sample Output

```
6
2.76
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q3. **Interface**

The Interface defines a rule that any classes that implement it should override all the methods. Let's implement Interface in our application. We'll start simple, by including display method in the Stall interface. Now all types of stalls that implement the interface should override the method.
Create an interface Stall with the following method
Create a class GoldStall which implements Stall interface with the following private attributes
Create default constructor and a parameterized constructor with arguments in order **GoldStall(String stallName, Integer cost, String ownerName, Integer tvSet)**.
Include appropriate getters and setters.
Include following methods
Create a class PremiumStall which implements Stall interface with following private attributes
Create default constructor and a parameterized constructor with arguments in order **PremiumStall(String stallName, Integer cost, String ownerName, Integer projector)**.
Include appropriate getters and setters.
Include following methods
Create a class ExecutiveStall which implements Stall interface with following private attributes
Create default constructor and a parameterized constructor with arguments in order **ExecutiveStall(String stallName, Integer cost, String ownerName, Integer screen)**.
Include appropriate getters and setters.
Include following methods

Input Format

The first line of the input consists of the choice. 1 for gold, 2 for premium and 3 for executive.
Next input is the stall details based on choice.

Output Format

The output prints the stall details. Refer sample input and output.

Sample Input

```
1
Mechanic
12000
Johnson
```

Sample Output

```
Mechanic 12000 Johnson 10
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q4. A group of ‘n’ candidates has applied for faculty recruitment. Their Name, qualification, experience, and gender are to be stored in the class “Recruitment”. Write a java program to sort the objects based on their experience and display their details.

If the experience is equal, use the name as the second sorting criterion.

Input Format

First line specifies the number of employees "n"
In the following lines Name, qualification, gender and experience of the faculty will be given for "n" employees

Output Format

Print the details of the faculty in the sorted order of their experience

Sample Input

```
2
ram
Be cse
male
```

Sample Output

```
pravin
Be ece
male
3
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q5. Write a java program that contains the class Doctor and class Patient. Compose the class Doctor in the class Patient. Class Doctor contains a constructor which sets the name, address, and passport no. Display the name of the person, address, and ward number class Patient.

Input Format

The first line of input consists of a name of a patient
The second line of input consists of an address of a patient
The third line of input consists of a ward number

Output Format

Output consists of a patient details

Sample Input

```
Raju
Chennai
34
```

Sample Output

```
Name : Raju
Address : Chennai
Ward Number : 34
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Answer Key & Solution

Section 1 - MCQ

Q1 class java.lang.Object

Solution

No Solution

Q2 any number of instances

Solution

No Solution

Q3 new

Solution

No Solution

Q4 Compilation Error

Solution

No Solution

Q5 If a class extends another class that has only one constructor that takes a value, then the child class must explicitly declare at least one constructor.

Solution

No Solution

Q6 Passing itself to method of the same class

Solution

No Solution

Q7 Increase concurrency and improve performance.

Solution

No Solution

Q8 Recursion

Solution

No Solution

Q9

Compilation error

Solution

No Solution

Q10

Yes, always

Solution

No Solution

Q11

translate() only

Solution

No Solution

Q12

Antarctica

Solution

No Solution

Q13

Both the mentioned

Solution

No Solution

Q14

At line B, constructor call

Solution

No Solution

Q15

150

Solution

No Solution

Q16

Compilation succeeds but at the time of creating object using default constructor, it throws compilation error

Solution

No Solution

Q17

I and III

Solution

No Solution

Q18

None of the above. We have a tie.

Solution

No Solution

Q19

Only the Duration class

Solution

No Solution

Q20

java.time.Instant class

Solution

No Solution

Section 2 - CODING

Q1

Test Case

Input	Output
	Dog Cat

Weightage - 100

Sample Input	Sample Output
	Dog Cat

Solution

Header

```
class Animal {

    void Print()
    {
        System.out.println("Animal");
```

```
    }
}


class Dog extends Animal {

    void Print()
    {
        System.out.println("Dog");
    }
}

class Cat extends Animal {

    void Print()
    {
        System.out.println("Cat");
    }
}
```

Footer

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

        Animal a;

        a = new Dog();
        a.Print();

        a = new Cat();
        a.Print();
    }
}
```

Q2

Test Case

Input

```
12
23
1.32
4 21
```

Output

```
276
5.5572
```

Weightage - 50

Input

```
76
21
3.211
43 21
```

Output

```
1596
138.74731
```

Weightage - 50

Sample Input

```
2
3
1.2
2 3
```

Sample Output

```
6
2.76
```

Solution

```
import java.util.*;
class Main
{
int Multiply(int a, int b)
{
    return a * b;
}

double Multiply(double a, double b)
{
    return a * b;
}
public static void main(String[] args)
{
    Scanner in=new Scanner(System.in);
    int a=in.nextInt();
    int b=in.nextInt();
    double c = in.nextDouble();
    double d=in.nextDouble();
    Main m=new Main();
    System.out.println(m.Multiply(a, b));
    System.out.println(m.Multiply(c, d));
 }
}
```

Q3

Test Case

Input

```
1
Mechanic
12000
Johnson
```

Output

```
Mechanic 12000 Johnson 10
```

Weightage - 20

Input

```
2
Knitting plaza
52000
Zain
```

Output

```
Knitting plaza 52000 Zain 4
```

Weightage - 20

Input

```
3
Fruit hunt
12000
Mahesh
```

Output

```
Fruit hunt 12000 Mahesh 8
```

Weightage - 20

Input

```
1
Book Expo
45000
Jegadheesh
```

Output

```
Book Expo 45000 Jegadheesh 8
```

Weightage - 20

Input

```
3
Vegetable hunt
45854
Alice
```

Output

```
Vegetable hunt 45854 Alice 7
```

Weightage - 20

Sample Input

```
1
Mechanic
12000
Johnson
```

Sample Output

```
Mechanic 12000 Johnson 10
```

Solution

```
import java.io.*;
import java.util.*;
interface Stall {
 void display();
}
class GoldStall implements Stall {
    private String stallName;
    private int cost;
    private String ownerName;
    private int tvSet;

    @Override
    public void display() {
        System.out.println(stallName+" "+cost+" "+ownerName+" "+tvSet);

    }

    public GoldStall() {
        this.stallName = null;
        this.cost = 0;
        this.ownerName = null;
        this.tvSet = 0;
    }

public GoldStall(String stallName, int cost, String ownerName, int tvSet) {
    this.stallName = stallName;
    this.cost = 0;
    this.ownerName = ownerName;
    this.tvSet = 0;
}

public String getStallName() {
    return stallName;
}

public void setStallName(String stallName) {
    this.stallName = stallName;
}

public int getCost() {
    return cost;
}

public void setCost(int cost) {
```

```
        this.cost = cost;
    }

    public String getOwnerName() {
        return ownerName;
    }

    public void setOwnerName(String ownerName) {
        this.ownerName = ownerName;
    }

    public int getTvSet() {
        return tvSet;
    }

    public void setTvSet(int tvSet) {
        this.tvSet = tvSet;
    }

}
class PremiumStall implements Stall {
    private String stallName;
    private int cost;
    private String ownerName;
    private int projector;
    public String getStallName() {
        return stallName;
    }
    public void setStallName(String stallName) {
        this.stallName = stallName;
    }
    public int getCost() {
        return cost;
    }
    public void setCost(int cost) {
        this.cost = cost;
    }
    public String getOwnerName() {
        return ownerName;
    }
    public void setOwnerName(String ownerName) {
        this.ownerName = ownerName;
    }
    public int getProjector() {
        return projector;
    }
    public void setProjector(int projector) {
        this.projector = projector;
    }
    public PremiumStall() {
        this.stallName = null;
        this.cost = 0;
        this.ownerName = null;
        this.projector = 0;
    }
    public PremiumStall(String stallName, int cost,String ownerName,int projector) {
        this.stallName = stallName;
        this.cost = cost;
        this.ownerName = ownerName;
        this.projector = projector;
    }
    @Override
    public void display() {
        System.out.println(stallName+" "+cost+" "+ownerName+" "+projector);
```

```
    }
}
class ExecutiveStall implements Stall {
    private String stallName;
    private int cost;
    private String ownerName;
    private int screen;
    public String getStallName() {
        return stallName;
    }
    public void setStallName(String stallName) {
        this.stallName = stallName;
    }
    public int getCost() {
        return cost;
    }
    public void setCost(int cost) {
        this.cost = cost;
    }
    public String getOwnerName() {
        return ownerName;
    }
    public void setOwnerName(String ownerName) {
        this.ownerName = ownerName;
    }
    public int getScreen() {
        return screen;
    }
    public void setScreen(int screen) {
        this.screen = screen;
    }
    public ExecutiveStall() {
        this.stallName = null;
        this.cost = 0;
        this.ownerName = null;
        this.screen = 0;
    }
    public ExecutiveStall(String stallName,int cost,String ownerName,int screen) {
        this.stallName = stallName;
        this.cost = cost;
        this.ownerName = ownerName;
        this.screen = screen;
    }
    @Override
    public void display() {
        System.out.println(stallName+" "+cost+" "+ownerName+" "+screen);

    }
}
class Main {
public static void main(String [] args) {
    int choice;
    Scanner sc = new Scanner(System.in);
    choice = Integer.parseInt(sc.nextLine());
    if(choice == 1) {
        GoldStall g = new GoldStall();
        g.setStallName(sc.nextLine());
        g.setCost(Integer.parseInt(sc.nextLine()));
        g.setOwnerName(sc.nextLine());
        g.setTvSet(Integer.parseInt(sc.nextLine()));
        g.display();
    }
    if(choice == 2) {
        PremiumStall p = new PremiumStall();
        p.setStallName(sc.nextLine());
```

```
            p.setCost(Integer.parseInt(sc.nextLine()));
            p.setOwnerName(sc.nextLine());
            p.setProjector(Integer.parseInt(sc.nextLine()));
            p.display();
        }
        if(choice == 3) {
            ExecutiveStall e = new ExecutiveStall();
            e.setStallName(sc.nextLine());
            e.setCost(Integer.parseInt(sc.nextLine()));
            e.setOwnerName(sc.nextLine());
            e.setScreen(Integer.parseInt(sc.nextLine()));
            e.display();
        }
    }
    }
```

Q4

Test Case

Input

```
3
ram
Be cse
male
```

Output

```
muzam
Be mechanical
male
5
```

Weightage - 10

Input

```
5
ram
Be cse
male
```

Output

```
surya
Be cse
male
5
```

Weightage - 20

Input

```
7
ram
Be cse
male
```

Output

```
priya
Be It
female
7
```

Weightage - 25

Input

```
10
ram
Be cse
male
```

Output

```
Imran
MCA
male
12
```

Weightage - 30

Input

```
7
ram
Be cse
male
```

Output

```
ram
Be cse
male
3
```

Weightage - 15

Sample Input

Sample Output

2 ram Be cse male	pravin Be ece male 3

Solution

```
import java.io.*;
import java.util.*;


class Recruitment implements Comparable<Recruitment>
{
    public String name, qualification, gender;
    public int experiance;

    public int compareTo(Recruitment m)
    {
        if (m.experiance - this.experiance == 0) {
            return m.name.compareTo(this.name);
        } else {
            return m.experiance - this.experiance;
        }
    }


    public Recruitment(String nm, String qua, String gender, int exp)
    {
        this.name = nm;
        this.experiance = exp;
        this.qualification = qua;
        this.gender = gender;
    }


    public String getName() {
        return name;
    }
}


class Main
{
    public static void main(String[] args)
    {
        ArrayList<Recruitment> emp_list = new ArrayList<Recruitment>();
        Scanner in = new Scanner(System.in);

        int num_of_emp;

        num_of_emp = Integer.parseInt(in.nextLine());

        for (int i=0;i<num_of_emp;i++) {
            int exp;
            String name, qua, gender;

            name = in.nextLine();
            qua = in.nextLine();
            gender = in.nextLine();
            exp = Integer.parseInt(in.nextLine());

            emp_list.add(new Recruitment(name, qua, gender, exp));
        }
```

```
        Collections.sort(emp_list);

        for (Recruitment each: emp_list)
        {
            System.out.println(each.name);
            System.out.println(each.qualification);
            System.out.println(each.gender);
            System.out.println(each.experiance);
        }
    }
}
```

Q5

Test Case

Input

```
Ganesh
Coimbatore
23
```

Output

```
Name : Ganesh
Address : Coimbatore
Ward Number : 23
```

Weightage - 100

Sample Input

```
Raju
Chennai
34
```

Sample Output

```
Name : Raju
Address : Chennai
Ward Number : 34
```

Solution

```
import java.util.Scanner;
class Doctor
{
    String Name;
    String Address;
    String Number;
    public Doctor(String Name,String Address,String Number)
    {
        this.Name = Name;
        this.Address = Address;
        this.Number = Number;
    }
}
class Main
{
    Doctor passport;
    Main(Doctor passport){
        this .passport = passport;
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        String name = sc.nextLine();
        String Address = sc.nextLine();
        String Number = sc.next();
        Doctor p = new Doctor(name,Address,Number);
        Main main = new Main(p);
        System.out.println("Name : " + main.passport.Name);
        System.out.println("Address : " + main.passport.Address);
        System.out.println("Ward Number : " + main.passport.Number);
```

```
    }
}
```