Class 10

Chapter – 12

Constructors

A. TICK (✓) THE CORRECT ANSWER

✅ A. Tick (✓) the correct answer

Q1. Which of the following is the correct statement?

a. A constructor is automatically called when an object is created
b. A single class can have one or more constructors
c. Constructor name must be same as class name
d. All of these

✅ **Answer: d. All of these**

💡 **Explanation:**
All statements are correct:

- Constructor runs automatically ✓
- Multiple constructors allowed ✓
- Name must match class ✓

Q2. Which of the following does not have a return type?

a. Method
b. Constructor
c. Both a and b
d. None

✅ **Answer: b. Constructor**

💡 **Explanation:**

- Methods have return type (even void)
- Constructor → NO return type

Q3. Which type of constructor is known as no-argument constructor?

a. Default
b. Parameterised

c. Non-parameterised
d. Copy

✅ **Answer: c. Non-parameterised**

💡 **Explanation:**
No-argument = constructor with **no parameters**

Q4. Example of constructor overloading

(From next page)

a. Multiple classes with same name
b. Multiple methods with same name
c. Multiple constructors with same name
d. Multiple constructors with same name but different parameters

✅ **Answer: d**

💡 **Explanation:**
Overloading = same name + different parameters

Q5. Output-based question

✓ Calls both constructors

```
this(10);
program(int x)
```

✅ **Answer: Parameterized constructor → Value of x = 10**

💡 Explanation:

- this(10) calls parameterized constructor
- Value becomes 10

Q6. Output of program

```
program obj1 = new program();
program obj2 = new program(10);
```

✅ **Answer:**
Default constructor
Parameterized constructor

💡 Explanation:

- obj1 → default
- obj2 → parameterized

Q7. What happens if no default constructor exists?

a. Object created
b. Compilation error
c. Runtime error
d. Default constructor created automatically

✅ **Answer: b. Compilation error**

💡 Explanation:
If only parameterized constructor exists → default not available

---

Q8. Constructor overloading true statement

a. Only one constructor
b. Cannot overload
c. Different parameter lists
d. Same parameter list

✅ **Answer: c**

💡 Explanation:
Overloading = different parameters

Q9. Keyword to call one constructor from another

a. super
b. this
c. self
d. constructor

✅ **Answer: b. this**

💡 Explanation:
this() is used to call another constructor

✅ B. Answer the following

Q1. What is a default constructor?

☞ A constructor with no parameters, provided automatically by Java if no constructor is defined.

Q2. Syntax to create object using constructor

ClassName obj = new ClassName();

Q3. Difference between parameterised & non-parameterised constructor

Non-parameterised	**Parameterised**
No arguments	Has arguments
Default values	User-defined values
Simple	Flexible

✓ C. Assertion & Reason

Q1

A: True
R: False

☞ **Answer: c**

💡 Overloading gives flexibility, not default initialization

Q2

A: True
R: False

☞ **Answer: c**

💡 Constructor has no return type (not even void)

Q3

A: True
R: False

☞ **Answer: c**

💡 Constructors must have same name (not different)

✅ D. Case-Based Questions

Q1. If constructor is not defined

a. Cannot create object
b. Compiler provides default constructor
c. Runtime provides
d. Abstract class

✅ **Answer: b**

💡 Java automatically gives default constructor

Q2. Parameterized constructor

a. No parameters
b. With parameters
c. Called once
d. Compiler provided

✅ **Answer: b**

Q3. Number of parameters

a. Must match variables
b. Can vary
c. Cannot exceed
d. Always zero

✅ **Answer: b**

Q4. Purpose of default constructor

a. Initialize variables
b. Overloading
c. Abstract class
d. Complex logic

✅ **Answer: a**

Q5. True about constructors

a. Only one constructor
b. Must define at least one

c. Multiple constructors allowed
d. Inherited

✅ **Answer: c**

✅ UNSOLVED PROGRAMS

◆ Program 1: Employee (Constructor)

```
import java.util.*;

class Employee {
   String ename;
   double basicsal, hra, da, pf, gross, net;

   Employee() {
      ename = "";
      basicsal = 0;
   }

   Employee(String n, double b) {
      ename = n;
      basicsal = b;
   }

   void calculate() {
      hra = 0.10 * basicsal;
      da = 0.55 * basicsal;
      pf = 0.0833 * basicsal;
      gross = basicsal + hra + da;
      net = gross - pf;
   }

   void display() {
      System.out.println(ename + " " + net);
   }

   public static void main(String args[]) {
      Employee e = new Employee("Rahul", 10000);
      e.calculate();
      e.display();
   }
}
```

◆ Program 2: Salesman

```
import java.util.*;

class Salesman {
   String sname;
   int sales;
```

```
    double rate, tot_amt;

    Salesman() {}

    void input() {
        Scanner sc = new Scanner(System.in);
        sname = sc.next();
        sales = sc.nextInt();
        rate = sc.nextDouble();
    }

    void compute() {
        tot_amt = sales * rate;
    }

    void display() {
        System.out.println(sname + " " + tot_amt);
    }
}
```

◆ Program 3: HCF & LCM

```
class HCF_LCM {
    int a, b, hcf, lcm;

    void input(int x, int y) {
        a = x; b = y;
    }

    void hcf_cal() {
        for(int i=1; i<=a && i<=b; i++) {
            if(a%i==0 && b%i==0)
                hcf = i;
        }
        System.out.println("HCF=" + hcf);
    }

    void lcm_cal() {
        lcm = (a*b)/hcf;
        System.out.println("LCM=" + lcm);
    }
}
```

◆ Program 4: Computer Price

```
class Computer {
    String model;
    double price, inc, total;

    Computer(String m, double p) {
        model = m;
        price = p;
    }

    void cal_price() {
```

```
        if(price <= 20000)
            inc = 0.025 * price;
        else if(price <= 30000)
            inc = 0.027 * price;
        else if(price <= 40000)
            inc = 0.04 * price;
        else
            inc = 0.06 * price;

        total = price + inc;
    }

    void display() {
        System.out.println(model + " " + total);
    }
}
```

◆ Program 5: Palindrome

```
class Palindrome {
    String word, rev="";

    void input(String w) {
        word = w;
    }

    boolean check() {
        for(int i=word.length()-1;i>=0;i--)
            rev += word.charAt(i);
        return word.equals(rev);
    }

    void display() {
        if(check())
            System.out.println("Palindrome");
        else
            System.out.println("Not Palindrome");
    }
}
```

◆ Program 6: Encode String

```
class Encode {
    String word, code="";

    Encode(String w) {
        word = w;
    }

    void convert() {
        for(int i=0;i<word.length();i++) {
            char ch = word.charAt(i);
            if(ch=='Z')
                code += 'A';
            else
```

```
            code += (char)(ch+1);
        }
        System.out.println(code);
    }
}
```

◆ Program 7: Pronic Number

```
class Pronic {
    int n;

    void input(int x) {
        n = x;
    }

    void check() {
        boolean flag = false;
        for(int i=0;i<n;i++) {
            if(i*(i+1)==n)
                flag = true;
        }

        if(flag)
            System.out.println("Pronic");
        else
            System.out.println("Not Pronic");
    }
}
```

◆ Program 8: Series

```
class Series {
    double sum=0, x, n;

    Series(double x1, double n1) {
        x = x1;
        n = n1;
    }

    void calculate() {
        for(int i=1;i<=n;i++)
            sum += Math.pow(x,i)/i;
    }

    void display() {
        System.out.println("Sum=" + sum);
    }
}
```

Q9. Class Pattern (Both Patterns)

◆ Pattern I

📃 Output:

```
1
12
123
1234
.....
12345...n
```

◆ Pattern II

📃 Output:

```
1
35
7911
.....
```

☞ Pattern explanation:

- Starts from **1**
- Increases by **2 (odd numbers)**

✓ Complete Java Program (Both Patterns in One Class)

```
class Pattern {
    int n;

    // Constructor
    Pattern(int x) {
        n = x;
    }

    // Pattern I
    void pattern1() {
        System.out.println("Pattern I:");

        for(int i = 1; i <= n; i++) {
            for(int j = 1; j <= i; j++) {
                System.out.print(j);
            }
            System.out.println();
        }
    }

    // Pattern II
    void pattern2() {
        System.out.println("\nPattern II:");

        int num = 1;

        for(int i = 1; i <= n; i++) {
            for(int j = 1; j <= i; j++) {
                System.out.print(num);
                num += 2;   // next odd number
            }
```

```
            System.out.println();
        }
    }

    public static void main(String args[]) {
        Pattern obj = new Pattern(4);  // change n here
        obj.pattern1();
        obj.pattern2();
    }
}
```

Q10. Bus Fare Program

Question Summary:

Calculate bus fare based on distance:

Distance	Rate
First 10 km	₹5 per km
Next 20 km	₹7.5 per km
Next 10 km	₹10 per km
Above that	₹15 per km

✓ Program:

```
import java.util.*;

class Fare {
    int dist;
    double bill;

    void input() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter distance: ");
        dist = sc.nextInt();
    }

    void calculate() {
        if(dist <= 10) {
            bill = dist * 5;
        }
        else if(dist <= 30) {
```

```
            bill = 10 * 5 + (dist - 10) * 7.5;
        }
        else if(dist <= 40) {
            bill = 10 * 5 + 20 * 7.5 + (dist - 30) * 10;
        }
        else {
            bill = 10 * 5 + 20 * 7.5 + 10 * 10 + (dist - 40) * 15;
        }
    }

    void display() {
        System.out.println("Total Fare = Rs. " + bill);
    }

    public static void main(String args[]) {
        Fare obj = new Fare();
        obj.input();
        obj.calculate();
        obj.display();
    }
}
```

****