**Experiment: 10**

PART A

(PART A: TO BE REFERRED BY STUDENTS)

**Aim:** **To study the Control statements in JAVA**

**Learning Outcomes: The learner would be able to**

**Theory:**

**Tasks:**

1. Explain and differentiate between for, while and do-while loop with appropriate example.
2. Write a menu-driven program until exit choice   
   1. Factorial  
   2. Pallindrome  
   3. Type casting (ascii)
3. Give output of following code:

public class Main{

public static void main(String [] args){

int a= 5;

int b= 10;

int c= 15;

int d= (++a) + 10 - (b--);

int e= b + c;

System.out.println("d: "+d);

System.out.println("e: "+ e);

}

}

1. Solve:

public class Main{

public static void main(String [] args){

int num;

System.out.println("Output: "+ (num>100 ? "H" : "L"));

}

}  
  
Predict Output for the following cases:  
1. num= 250

2. num= -100

3. num= 97

4. num= 1001

5. num= -200

1. Differentiate the following:
   1. Constructors and Methods
   2. Default and non parameterized constructor
2. Constructor overloading is also one form of polymorphism. If Yes explain?
3. What is the difference between final, finally, and finalize in Java?
4. What is the difference between abstract class and concrete class? And when do we use the Abstract method in Java?

**PART B**

(PART B: TO BE COMPLETED BY STUDENTS)

Students must execute all the tasks in Experiment-1 and copy paste the code, along with the snapshot of the output in Part-B. Upload and Submit the Part-B in soft copy on the portal. The filename should be **OOPJ\_batch\_rollno\_experimentno Example: OOPJ\_A1\_A001\_P1**

|  |  |
| --- | --- |
| **Roll No.: C126** | **Name: Rushabh Abhay Shah** |
| **Prog/Yr/Sem: BTI/4/8** | **Batch: 2021-2027** |
| **Date of Experiment: 05-04-2025** | **Date of Submission: 05-04-2025** |

1. Explain and differentiate between for, while, and do-while loop with appropriate examples.

🡪

**For Loop**

Purpose: Used when the number of iterations is known.

Syntax: for (initialization; condition; increment/decrement) { code }

Example:

for (int i = 0; i < 5; i++) {

System.out.println(i);

}

**While Loop**

Purpose: Used when the number of iterations is not known.

Syntax: while (condition) { code }

Example:

int i = 0;

while (i < 5) {

System.out.println(i);

i++;

}

**Do-While Loop**

Purpose: Similar to while loop but executes at least once.

Syntax: do { code } while (condition);

Example:

int i = 0;

do {

System.out.println(i);

i++;

} while (i < 5);

**2. Write a menu-driven program until exit choice**

**🡪**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("1. Factorial");

System.out.println("2. Palindrome");

System.out.println("3. Type Casting (ASCII)");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.print("Enter a number for factorial: ");

int num = scanner.nextInt();

System.out.println("Factorial of " + num + " is " + factorial(num));

break;

case 2:

System.out.print("Enter a string to check palindrome: ");

scanner.nextLine(); // Consume newline left-over

String str = scanner.nextLine();

System.out.println(str + " is " + (isPalindrome(str) ? "" : "not ") + "a palindrome.");

break;

case 3:

System.out.print("Enter a character for ASCII value: ");

char ch = scanner.next().charAt(0);

System.out.println("ASCII value of '" + ch + "' is " + (int) ch);

break;

case 4:

System.out.println("Exiting...");

break;

default:

System.out.println("Invalid choice. Please try again.");

}

} while (choice != 4);

}

static int factorial(int n) {

if (n == 0 || n == 1)

return 1;

else

return n \* factorial(n - 1);

}

static boolean isPalindrome(String s) {

int i = 0, j = s.length() - 1;

while (i < j) {

if (s.charAt(i) != s.charAt(j))

return false;

i++;

j--;

}

return true;

}

}

**3. Give output of the following code:**

🡪

public class Main {

public static void main(String[] args) {

int a = 5;

int b = 10;

int c = 15;

int d = (++a) + 10 - (b--);

int e = b + c;

System.out.println("d: " + d);

System.out.println("e: " + e);

}

}

Output:

d = (++a) + 10 - (b--)

d = 6 + 10 - 10

d = 6

e = 9 + 15 = 24.

**4. Solve and predict output for the following code:**

public class Main {

public static void main(String[] args) {

int num;

System.out.println("Output: " + (num > 100 ? "H" : "L"));

}

}

Output for each:

num = 250: Output will be "H".

num = -100: Output will be "L".

num = 97: Output will be "L".

num = 1001: Output will be "H".

num = -200: Output will be "L".

**5. Differentiate between:**

**a) Constructors and Methods**

Constructors: Special methods used to initialize objects when they are created. They have the same name as the class and no return type (not even void).

Methods: Used to perform specific actions. They can return values and have a return type.

**b) Default and non-parameterized constructor**

Default Constructor: A constructor with no parameters that is automatically provided by Java if no other constructor is defined.

Non-Parameterized Constructor: A constructor with no parameters that is explicitly defined by the programmer.

**6. Is constructor overloading a form of polymorphism? If yes, explain.**

Yes, constructor overloading is a form of polymorphism. Polymorphism allows objects of different classes to be treated as objects of a common superclass. Constructor overloading is a specific case of compile-time polymorphism where multiple constructors with different parameter lists can be defined for a class, allowing objects to be initialized in different ways.

**7. What is the difference between final, finally, and finalize in Java?**

final: Used to restrict modification. Can be applied to variables (constant), methods (cannot be overridden), and classes (cannot be subclassed).

finally: Used with try-catch blocks to ensure that certain code is executed regardless of whether an exception is thrown.

finalize: A method that is called by the garbage collector before an object is destroyed. It is used for cleanup operations.

**8. What is the difference between an abstract class and a concrete class? And when do we use the Abstract method in Java?**

Abstract Class: Cannot be instantiated and is used to provide a partial implementation that can be shared by subclasses. It can contain both abstract and concrete methods.

Concrete Class: A complete class that can be instantiated and does not contain any abstract methods.

Abstract methods are used when you want to define a method that must be implemented by any subclass. They are declared in abstract classes and have no implementation.