Java Programming Assignment

Section 1: Java Data Types

1. What are the different primitive data types available in Java?
   Java has 8 primitive data types, each designed for storing simple values directly in memory.

Data Type	Size (bits)	Default Value	Description
byte	8	0	Stores whole numbers from -128 to 127.
short	16	0	Stores whole numbers from -32,768 to 32,767.
int	32	0	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	64	0L	Stores large whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
float	32	0.0f	Stores decimal numbers (single-precision floating point).
double	64	0.0d	Stores decimal numbers (double-precision floating point.
char	16	'\u0000'	Stores a single character or Unicode symbol.
boolean	1	false	Stores one of two values: true or false.

2. Explain the difference between primitive and non-primitive data types in Java.

   Primitive type: Built-in types that store actual values directly in memory.

   Ex: byte, short, int, long, float, double, char, Boolean

   Non-primitive type: Also known as reference types, these store references (addresses) to objects, not the actual data itself.

   Ex: strings, arrays, classes, interfaces, objects, collections

3. Write a Java program that demonstrates the use of all primitive data types.

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

      byte bytevar = 100;
      System.out.println("byte value: " + bytevar);

      short shortvar = 30000;
      System.out.println("short value: " + shortvar);

      int intvar = 100000;
      System.out.println("int value: " + intvar);

      long longvar = 10000000000L;
      System.out.println("long value: " + longvar);

      float floatvar = 3.14f;
      System.out.println("float value: " + floatvar);

      double doublevar = 3.14159265359;
      System.out.println("double value: " + doublevar);

      char charvar = 'A';
      System.out.println("char value: " + charVar);

      boolean boolvar = true;
      System.out.println("boolean value: " + boolvar);
   }
}
```

Output:
byte value: 100
short value: 30000
int value: 100000
long value: 10000000000
float value: 3.14
double value: 3.14159265359
char value: A
boolean value: true

4. What is type casting? Provide an example of implicit and explicit casting in Java.

   Type casting in Java is the process of converting a variable from one data type to another.

   Implicit(Widening)Casting: Done automatically by Java when converting smaller data types to larger compatible types.
   Ex: int->long

```
public class ImplicitExample {
   public static void main(String[] args) {
      int intNum = 100;      // 32-bit
      double doubleNum = intNum;
      System.out.println("Implicit casting (int to double): " + doubleNum);
   }
}
```

   Output: Implicit casting (int to double): 100.0

   Explicit(Narrowing)Casting: Manually done by the programmer when converting larger data types to smaller types.
   Ex: double->int

```
public class ExplicitExample {
   public static void main(String[] args) {
      double doubleNum = 9.78;
      int intNum = (int) doubleNum;
      System.out.println("Explicit casting (double to int): " + intNum);
   }
}
```

   Output: Explicit casting (double to int): 9

5. What is the default value of each primitive data type in Java?

Default Values of Primitive Data Types

Data Type	Default Value
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000' (null character)
boolean	false

Section 2: Java Control Statements

1. What are control statements in Java? List the types with examples.
In Java, control statements are instructions that control the flow of execution in a program, deciding which code runs, how many times, and in what order.

Types of Control Statements in Java

1. Decision-Making Statements
Used to execute certain code blocks based on conditions.

a) if statement

```
int age = 18;
if (age >= 18) {
  System.out.println("You are an adult.");
}
```

Output:You are an adult

b) if-else statement

```
int number = 5;
if (number % 2 == 0) {
  System.out.println("Even");
} else {
  System.out.println("Odd");
}
```

Output:Odd

c)if-else-if ladder

```
int marks = 75;
if (marks >= 90) {
   System.out.println("Grade A");
} else if (marks >= 75) {
   System.out.println("Grade B");
} else {
   System.out.println("Grade C");
}
```

Output: Grade B

d)switch statement

```
int day = 3;
switch (day) {
   case 1: System.out.println("Monday"); break;
   case 2: System.out.println("Tuesday"); break;
   case 3: System.out.println("Wednesday"); break;
   default: System.out.println("Invalid day");
}
```

Output: Wednesday

2)Looping statements

Used to execute a block of code repeatedly.

a)for loop

```
  for (int i = 1; i <= 5; i++) {
   System.out.println(i);
}
```

Output: 1
2
3
4
5

b)while loop

```
int i = 1;
while (i <= 5) {
   System.out.println(i);
   i++;
}
```

Output: 1
2
3

4
5

c)do-while loop

```
int j = 1;
do {
   System.out.println(j);
   j++;
} while (j <= 5);
```

Output: 1
2
3
4
5

d)for-each loop

```
int[] numbers = {10, 20, 30};
for (int num : numbers) {
   System.out.println(num);
}
```

Output: 10
20
30

3)Jump statements

Used to alter the normal flow of control.

a)break

```
for (int i = 1; i <= 5; i++) {
   if (i == 3) break;
   System.out.println(i);
}
```

Output: 1
2

b)continue

```
for (int i = 1; i <= 5; i++) {
   if (i == 3) continue;
   System.out.println(i);
}
```

Output: 1
2
4
5

c)return

```
public class ReturnExample {
   public int add(int a, int b) {
      return a + b;
   }

   public static void main(String[] args) {
      ReturnExample obj = new ReturnExample ();

      int result = obj.add(5, 7);
      System.out.println("Sum: " + result);
   }
}
```

Output: Sum:12

2. Write a Java program to demonstrate the use of if-else and switch-case statements.

```
import java.util.Scanner;

public class IfElseSwitchDemo {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      System.out.print("Enter your marks (0-100): ");

      int marks = sc.nextInt();

      if (marks >= 50) {

         System.out.println("Result: Pass ");

      } else {

         System.out.println("Result: Fail ");

      }

      System.out.print("Enter a day number (1-7): ");

      int day = sc.nextInt();

      switch (day) {
```

```
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
    case 3:
        System.out.println("Wednesday");
        break;
    case 4:
        System.out.println("Thursday");
        break;
    case 5:
        System.out.println("Friday");
        break;
    case 6:
        System.out.println("Saturday");
        break;
    case 7:
        System.out.println("Sunday");
        break;
    default:
        System.out.println("Invalid day number!");
}

sc.close();
```

```
    }
}
```

Output:

Enter your marks (0-100): 75

Result: Pass

Enter a day number (1-7): 3

Wednesday

3. What is the difference between break and continue statements?
   break Statement:
   Immediately terminates the loop or switch it is inside.
   Control jumps out of the loop to the first statement after it.

   continue Statement:
   Skips the rest of the loop for the current iteration and moves to the next iteration.
   Loop continues running, but the skipped iteration's code is not executed.

4. Write a Java program to print even numbers between 1 to 50 using a for loop.

```
public class Evennumbers {
    public static void main(String[] args) {
        System.out.println("Even numbers between 1 and 50:");

        for (int i = 1; i <= 50; i++) {
            if (i % 2 == 0) {
                System.out.print(i + " ");
            }
        }
    }
}
```

   Output:
   Even numbers between 1 and 50:
   2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50

5. Explain the differences between while and do-while loops with examples.
   while Loop:
   Condition checked first → body executes only if the condition is true.
   Might not execute at all if the condition is false from the start.

Ex:

```
int i = 1;
while (i <= 3) {
   System.out.println(i);
   i++;
}
```

Output:1
2
3

do-while Loop:
Body executes first, then condition is checked.
Executes at least once, even if the condition is false.
Ex:

```
int j = 1;
do {
   System.out.println( j);
   j++;
} while (j <= 3);
```

Output:1
2
3

Section 3: Java Keywords and Operators

1. What are keywords in Java? List 10 commonly used keywords.
   Keywords in Java are reserved words that have a predefined meaning in the language.
   we cannot use them as variable names, method names, class names, or identifiers.
   10 Commonly Used Java Keywords
   Class, public, static, void, int, if, else, for, return, new.

2. Explain the purpose of the following keywords: static, final, this, super.
   Static:
   Belongs to the class rather than an instance (object).
   Shared among all objects of that class.
   Can be used for variables, methods, blocks, and nested classes.

   Final:
   Used to declare constants, prevent method overriding, or prevent class inheritance.

   This:
   Refers to the current object.
   Used to access instance variables and methods of the current class.

Super:
Refers to the parent class (superclass) object.
Used to call:
Parent class constructors.
Parent class methods that are overridden.

3. What are the types of operators in Java?
   operators are special symbols that perform operations on variables and values.
   Types of operators in java:
   Arithmetic operators: + , - , *, / ,%
   Relational operators: ==, != , <, >, <= , >=
   Logical operators: &&, || , !
   Assignment operators : =, +=, -=, *=,/=, %=
   Bitwise operators: &,| ,^ , ~, << , >>
   Unary operators: + , -, ++, --,!,~

4. Write a Java program demonstrating the use of arithmetic, relational, and logical operators.

```
public class OperatorsDemo {
  public static void main(String[] args) {
    int a = 10, b = 5;
    System.out.println("Arithmetic Operators:");
    System.out.println("a + b = " + (a + b));
    System.out.println("a - b = " + (a - b));
    System.out.println("a * b = " + (a * b));
    System.out.println("a / b = " + (a / b));
    System.out.println("a % b = " + (a % b));

    System.out.println("\nRelational Operators:");
    System.out.println("a == b: " + (a == b));
    System.out.println("a != b: " + (a != b));
    System.out.println("a > b : " + (a > b));
    System.out.println("a < b : " + (a < b));
    System.out.println("a >= b: " + (a >= b));
    System.out.println("a <= b: " + (a <= b));

    boolean x = true, y = false;
    System.out.println("\nLogical Operators:");
    System.out.println("x && y: " + (x && y));
    System.out.println("x || y: " + (x || y));
    System.out.println("!x    : " + (!x));
  }
```

```
}
```

Output:
Arithmetic Operators:
a + b = 15
a - b = 5
a * b = 50
a / b = 2
a % b = 0

Relational Operators:
a == b: false
a != b: true
a > b : true
a < b : false
a >= b: true
a <= b: false

Logical Operators:
x && y: false
x || y: true
!x : false

5. What is operator precedence? How does it affect the outcome of expressions?

   Operator precedence in Java refers to the rules that determine the order in which different operators in an expression are evaluated.
   When you have multiple operators in a single expression, Java doesn't just go left to right — it follows a precedence hierarchy to decide which operation happens first.

   Ex:
   ```
   int result = 10 + 5 * 2;
   System.out.println(result);
   ```

   Output:
   20