Worksheet -3

Q1. Which one of the following is not a Java feature?
A. Object-oriented B. Use of pointers C. Portable D. Dynamic and Extensible

Pointers is not a Java feature. Java provides an efficient abstraction layer for developing without using a pointer in Java. Features of Java Programming are Portable, Architectural Neutral, Object-Oriented, Robust, Secure, Dynamic and Extensible, etc.
So the output is **B. Use of pointers**

Q2. Which of these cannot be used for a variable name in Java?
A. identifier & keyword B. identifier C. keyword D. none of the mentioned

Keywords are specially reserved words which can not be used for naming a user defined variable, example: class, int, for etc.
So the output is **C. keyword**

Q3.Which of the following is a superclass of every class in Java?
A. ArrayList B. Abstract class C. Object class D. String

The Object class is the superclass of all other classes in Java and a part of the built-in java.
So the output is **C.Object class**

Q4.Which one is a valid declaration of a boolean?
A. boolean b1 = 1; B. boolean b2 = 'false';
C. boolean b3 = false; D. boolean b4 = 'true'

Valid declaration for Boolean is boolean b3 = false;. "Boolean" is a "data type" that can either be True or False.
So the output is **C. boolean b3 = false;**

Q5. Which is the modifier when there is none mentioned explicitly?
A. protected B. private C. public D. default

Default is the access modifier when none is defined explicitly. It means the member (method or variable) can be accessed within the same package.

Q6.All the variables of interface should be? A. default and final B. default and static C. public, static and final D. protect, static and final

Variables of an interface are public, static and final by default because the interfaces cannot be instantiated, final ensures the value assigned cannot be changed with the implementing class and public for it to be accessible by all the implementing classes.
So the output is **C. public, static and final**

Q7.Which of these data types is used to store command line arguments?
A. Array B. Stack C. String D. Integer

All command Line arguments are passed as a string.
So the output is **C. String**

Q8.How many arguments can be passed to main()?
A. Infinite B. Only 1 C. System Dependent D. None of the mentioned

The main method can accept a single parameter: an array of String. While the main method itself only takes one argument (the array), this array can contain any number of string elements.
So the output is **B. Only 1**

Q9.What will be the output of the following Java program, Command line execution is done as – "java Output This is a command Line"? class Output { public static void main(String args[]) { System.out.print(args[0]); } }
A. java B. Output C. This D. is

When the command java output This is a command Line is executed,
args[0] is "This"
args[1] is "is"
args[2] is " command"
args[3] is " Line"
So the output is **C. This**

Q10.What is the value of "d" in the following Java code snippet? double d = Math.round (2.5 + Math.random()); A. 2 B. 3 C. 4 D. 2.5

2.5 + Math.random()
This expression will result in a value between 2.5 and 3.5 (since adding 0.0 to 2.5 gives 2.5 and adding 1.0 to 2.5 gives 3.5)

Math.round(2.5 or3.5) will be 3.0
So the output is **B. 3**

Q11.Which of these methods is a rounding function of Math class?
A. max() B. min() C. abs() D. all of the mentioned

max(), min(), abs() are all rounding functions of Math class.
Examples :
- max() – Math.max(3,5) , output is 5
- min() – Math.min(3,5) , output is 3
- abs() – Math.abs(-4.7) , output is 4.7

So the output is **D.all of the mentioned**

Q12.Standard output variable 'out' is defined in which class?
A. Void B. Process C. Runtime D. System

Variable "out" is defined in System class
When we try to print something in java like —- System.out.print("hello world");
So the output is **D. System**

Q13.What will be the output of the java program?
class main_class { public static void main(String args[]) { int x = 9; WORKSHEET if (x == 9) { int x = 8; System.out.println(x); } } }
A. 9 B. 8 C. Compilation error D. Runtime error

In the given code of java, x is defined with value 9 , after checking if condition is correct or not x has value 9 and comparing value is also 9 so the if condition is true.
There is an error which is x is already defined above to we cannot re-initialize the variable So the output is **C.Compilation error**

Q14.Which of these is the method which is executed first before execution of any other thing takes place in a program? A. main method B. static method C. private method D. finalize method

If a static method is present in the program then it will be executed first, then main will be executed.
So the output is **B. static method**

Q15.Which of these can be used to differentiate two or more methods having the same name? A. Parameters data type B. Number of parameters C. Return type of method D. All of the mentioned

Return type of method, Number of parameters and Parameters data type can be used to differentiate two or more methods having the same name.

Q16. What will be the output of the following Java program? class Output { static void main(String args[]) { int x , y = 1; x = 10; if(x != 10 && x / 0 == 0) System.out.println(y); else System.out.println(++y); } } A. 1 B. 2 C. Runtime Error D. Compilation Error

In a logical AND operation, if the first condition is false, the second condition is not evaluated because the overall result cannot be true.
The first condition x!=10 is evaluated. Since x is 10, `x != 10` is `false`.
Because the first condition is `false`, the second condition `x / 0 == 0` is not evaluated. This is due to short-circuit evaluation which prevents a division by zero error.

Since the `if` condition is false, the `else` block is executed.
`System.out.println(++y);` increments y by 1 before printing. So, y becomes 2.

Q17.What will be the output of the following Java program? class area { int width; int length; int height; area() { width = 5; length = 6; height = 1; } void volume() { volume = width * height * length; } } class cons_method { public static void main(String args[]) { area obj = new area(); obj.volume(); System.out.println(obj.volume); } } A. 0 B. 1 C. 25 D. 30

The `Area` class has instance variables `width`, `length`, `height`, and `volume`.
The constructor `Area()` initializes `width`, `length`, and `height`.
The `Volume()` method calculates the volume and stores it in the instance variable `volume`.
In the `ConsMethod` class, the `main` method creates an `Area` object, calls the `Volume()` method, and prints the volume.
So the output is **D. 30**

Q18. Write Syntax to create/define java methods.

// Access Modifier: Determines the visibility of the method (e.g., public, private, protected, default).

```
// Return Type: Specifies the type of value the method returns. Use 'void' if the method does not return a value.
// Method Name: The name of the method, following the naming conventions.
// Parameters: Optional, enclosed in parentheses. Parameters are variables that accept values passed to the method.

[Access Modifier] [Return Type] [Method Name]([Parameters]) {
    // Method body: Code to be executed when the method is called.
    // This can include declarations, expressions, and statements.
}

// Example 1: A simple method without parameters and return value
public void exampleMethod() {
    // Method body
    System.out.println("This is a simple method.");
}

// Example 2: A method with parameters and a return value
public int add(int a, int b) {
    // Method body
    int sum = a + b;
    return sum; // Return statement
}

// Example 3: A private method that performs a specific task and does not return a value
private void displayMessage(String message) {
    // Method body
    System.out.println(message);
}

// Example 4: A protected method that returns a double value and takes no parameters
protected double calculateArea(double radius) {
    // Method body
    return Math.PI * radius * radius;
}
```

Components of a Method:

1. **Access Modifier:**
    - `public`: The method is accessible from any other class.

    - `private`: The method is accessible only within the class it is defined.
    - `protected`: The method is accessible within the same package and subclasses.
    - Default (no modifier): The method is accessible within the same package.
2. **Return Type:**
    - Specifies the type of value the method returns. If the method does not return any value, use `void`.
3. **Method Name:**
    - The name of the method should be a valid identifier and follow naming conventions (camelCase).
4. **Parameters:**
    - Enclosed in parentheses, parameters are optional and can include one or more variables separated by commas. Each parameter has a type and a name.
5. **Method Body:**
    - Enclosed in curly braces `{}`, this part contains the code to be executed when the method is called. It can include declarations, expressions, statements, and return statements if applicable.

Q19. Write a java program following instructions A. Make a class Addition a. initialize sum as 0 WORKSHEET b. make addTwoInt method taking two int parameters a,b. make sum = a+b. Return Sum B. define class as Method Call. Define main method a. Create object of class Addition b. call method using instance of object c. Print sum

```
// Class Addition
class Addition {
    int sum = 0; // Initialize sum as 0

    // Method to add two integers
    public int addTwoInt(int a, int b) {
        sum = a + b; // Calculate sum
        return sum; // Return sum
    }
}

// Class MethodCall
public class MethodCall {
```

```
    public static void main(String[] args) {
        // Create an object of the class Addition
        Addition additionObj = new Addition();

        // Call the addTwoInt method using the instance of the object
        int result = additionObj.addTwoInt(5, 10);

        // Print the sum
        System.out.println("Sum: " + result);
    }
}
```

Output :: Sum: 15

Q20. Write a java program following instructions A. Define a class Example a. Define two instance variables number and name b. Define accessor (getter) methods c. Define mutator (setter) methods d. define method printDetails —-> print name and number B. Define public class Demo (Main Class) a. Define main method b. Make Instance/object of example class c. set number and name using instance created as 123 and Your name. d. call printDetails method using instance

```
// Class Example
class Example {
    private int number; // Instance variable for number
    private String name; // Instance variable for name

    // Accessor (getter) method for number
    public int getNumber() {
        return number;
    }

    // Mutator (setter) method for number
    public void setNumber(int number) {
        this.number = number;
    }

    // Accessor (getter) method for name
    public String getName() {
        return name;
    }
```

```
    // Mutator (setter) method for name
    public void setName(String name) {
        this.name = name;
    }

    // Method to print details
    public void printDetails() {
        System.out.println("Name: " + name);
        System.out.println("Number: " + number);
    }
}

// Public class Demo (Main Class)
public class Demo {
    public static void main(String[] args) {
        // Create an instance/object of the Example class
        Example exampleObj = new Example();

        // Set number and name using the instance created
        exampleObj.setNumber(123);
        exampleObj.setName("Your name");

        // Call the printDetails method using the instance
        exampleObj.printDetails();
    }
}
```

Output :

```
Name: Your name
Number: 123
```