24/7/23

ASSIGNMENT 1

PAGE NO.: 3

Q.1 Write difference between Procedure Oriented Programming and Object Oriented Programming.

→

POP	OOP
Program is divided into functions	Program is divided into objects
Top - down approach	Bottom - up approach
Expanding new data and functions is not easy	Adding new data and functions is easy.
It doesn't use access specifier	It uses access specifier
Data is often passed as arguments to functions and ~~they~~ functions operate on this data	Objects hold ~~state~~ data & expose methods to interact with that data.
No data hiding	Encapsulation is used to hide the data.
Inheritance is not allowed.	Inheritance property is used.
eg: C, Pascal	eg: Java, Python.

Q.2 Enlist features of Java. Explain any four
—> i) Platform Independence
> Java's platform independence is achieved through JVM. When you compile a Java source code, it is converted into bytecode, ~~and~~ so that it can then run on any system that has a compatible JVM. Provides portability and reduces the need to rewrite code.

ii) Object - Oriented Programming Support.
> Java's OOP support enables developers to create modular, reusable and well-structured code. Concepts like inheritance, polymorphism enables ~~objects~~ dynamic method binding and flexibility in code

iii) Garbage Collection
> Java's garbage collection mechanism automates memory ~~all~~ management, reducing the burden on developers to manually allocate and deallocate memory. Also it prevents memory leaks and helps ~~ensure~~ in ensuring stable and efficient program executions

iv) Robust and Secure.
> Java's robustness is a result of features like exception handling and run time error checking which catches error

early and it makes easier for developer to fix it. Also, Java's security model ensures that untrusted code runs in a controlled environment, reduces the risk of malicious activity on the host system.

v) Multi-threading Support
vi) Rich Standard library
vii) High Performance
viii) Community and Ecosystem

Q.3 JVM is platform dependent. Justify.
—ᵛ The main purpose of the JVM is to execute Java bytecode. However, the JVM does not execute bytecode directly, instead, it translates it into native machine code so that JVM is implemented differently for each target platform (eg: windows, macOS, linux). The JVM uses native libraries and APIs to interact with the underlying operating system, which handles the tasks like memory management, I/O operations, threading and other.

While Java's "write once, run anywhere" motto emphasizes the portability of Java source code, it relies on the presence of a platform specific JVM on each target platform to achieve this portability.

Q.4 Explain Data encapsulation & Data abstraction

—↓

i) Data Encapsulation

↳ This OOP concept promotes data hiding and protection. It is the process of bundling data and methods that operates on that data within a single unit called class. Basically the data members of a class are declared as private or protected, preventing direct acess or modification from outside the class.

ii) Data Abstraction

↳ This OOP concept focuses on providing a simplified and generalized view of objects or classes. It allows complex data structures, operations or behaviours to be represented in a more user friendly and understandable way, by exposing only the relevant details and hiding unnecessary complexities.

Q.5 Explain API, JDK & IDE.

—↓

i) API

↳ Application Programming Interface, is a set of rules and protocols that allows different software applications to communicate and interact with each other. It enables applications to request data from other applications without needing to understand their internal implementation.

ii) JDK

→ Java Development Kit, is a software development kit provided by Oracle that includes a collection of tools, executables and libraries necessary for developing Java applications and it also provide everything needed to compile, run and debug Java Programs

iii) IDE

→ Integrated Development Environment is a software application that provides comprehensive tools and features to streamline the process of software development. It basically integrates a text editor, a compiler, a debugger to make it easier for developers to write, test and debug code

Q.6 Explain defining and calling method.

→ Defining a Method: It involves declaring its signature, specifying its behavior or code block and optionaly defining parameter and return type (output).

Syntax: returnType methodName (parameterType parameterName, ...)

```
{
    // code
}
```

eg:

eg: public int add(int n1, int n2) # method define
{
    int sum = n1 + n2;
    return sum;
}

Calling a Method : Once a method is defined you can call it from other parts of the program to execute the code within the method. When calling a method, you typically provide arguments that match the parameter type defined in method's signature.
To call a method, write its name followed by parentheses.

eg: public static void main (String args[])
{
    int a = 5;
    int b = 10;
    int result = add(a,b); # method call
    System.out.println("The sum of " + a + "and" + b + "is :" + result);
}

Q.7 Explain public static void main (String args[]) in detail.

-> i) public : It is an access specifier which indicates that the main method can be accessed from outside the class.

ii) static : It is a keyword that indicates that the main method belongs to the class itself, rather than to an instance of the class.

iii) void : It is the return type of the main method. It means that the main method does not return any value

iv) main : The name of the method.

v) String args[] : The parameter of the main method which is an array of Strings that allows you to pass command line arguments.

Q.8 Explain type conversion with example.
→ Type conversion is the process of converting one data type into another. There are two main types of type conversion :

i) Implicit Type Conversion
* Conversion where the programming language automatically converts one data type to another without the need for the programmer to forcefully specify the conversion. Usually takes place when there is no risk of data loss

eg: int num = 11;
double num_dd = num; # Implicit

ii) Explicit Type Conversion
> Conversion where the programmer explicitly specifies the conversion from one data type to another. Typically used when there is a possibility of data loss.

eg: double numDouble = 10.5;
int num = (int) numDouble; #Explicit

Q.9 Explain scope of variable with suitable example.
→ The scope of variable defines the region of the program where the variable is accessible and can be used.

i) Local Scope
> A variable declared within a specific block of code and can only be accessed within that block and is not visible outside.

```
eg: public class Test
    {
        void method1()
        {
            int x;
        }
    }
```

ii) Instance Scope
> A variable declared within a class but

outside any method. They are associated with objects of the class and have distinct values for each object.

iii) Class Scope
→ A variable declared inside a class and it can be accessed anywhere in class.

eg:
```
public class Scope
{
    String name = "Tony"; #instance variable
    Static double height = 5.9; #class variable
    public static void main (String args[])
    {
        int marks = 97; #local variable
    }
}
```

Q.10 Explain one dimensional array & write a program that creates and initializes a four integer element array. Calculate and display the average of its values.

→ A one dimensional array is a collection of elements of the same type, arranged in a linear sequence. Each element in the array is identified by its index, starting from '0' for the first element. It provides a convenient way to access and manipulate multiple elements using a single variable.

Program:

```
public class Average
{
    public static void main (String args[])
    {
        int [] n = {10, 20, 30, 40};
        int sum = 0;
        for (int i = 0; i < n.length; i++)
        {
            sum += n[i];
        }
        double average = (double) sum / n.length;
        System.out.println ("Array Elements: ");
        for (int i = 0; i < n.length; i++)
        {
            System.out.println (n[i] + " ");
        }
        System.out.println();
        System.out.println ("Average: " + average);
    }
}
```

Output: Array Elements: 10 20 30 40
Average: 25.0

Q.11 Explain break and continue statement with example.

—

i) Break Statement.

* Used to immediately terminate the execution of a loop. When 'break' is encountered it jumps to the next statement

after the loop or switch block.

eg:
```
for (int i=0 ; i<=5 ; i++)
{
    if (i==3)
    {
        break;
    }
    System.out.println("i = " + i);
}
```

# Output : i = 0
i = 1
i = 2

ii) Continue Statement
> Used to skip the rest of the current iteration of a loop and continue with the next iteration.

eg:
```
for (int i=0 ; i<=5 ; i++)
{
    if (i==3)
    {
        continue;
    }
    System.out.println( i);
}
```

Output : 0 1 2 4 5

Q.12 Explain instanceof and Ternary operator.

→ i) Instanceof Operator

* Used to test whether an object is an instance of a particular class or implements a specific interface. It returns a boolean value based on whether the object is of specified type or a subtype of that type.

- Syntax: object instanceof type

ii) Ternary Operator

* Conditional operator, is a shorthand way to write simple if-else statements. It takes three operands: a condition, an expression to be evaluated if the condition is true and an expression to be evaluated if the condition is false.

- Syntax: condition ? expression1 : expression2

Q.13 Difference between while and do-while loop.

→ Both are loop structures that allow you to execute a block of code repeatedly based on a specified condition.

i) While loop

* It is a entry controlled loop, meaning

that it first checks the loop condition before executing the code inside the loop. The loop will run / execute as long as the condition is remains 'true'.

- Syntax: while (condition)
  {
  // Code to be executed repeatedly
  }

ii) Do-while loop

* It is a exit controlled loop, meaning that it first executes the code inside the loop and then checks the loop condition. This guarantees that the code inside the loop will be executed at least once, regardless of the condition's initial value.

- Syntax: do
  {
  // Code to be executed repeatedly
  } while (condition);

Q.14 Write a program which takes five numbers as command line arguments from user, store them in one dimensional array and display count of negative numbers.

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

```
{
    if (args.length != 5)
    {
        System.out.println("Enter exactly
        five numbers");
        return;
    }
    int[] num = new int[5]
    for (int i=0; i<5; i++)
    {
        num[i] = Integer.parseInt(args[i]);
    }
    int count = 0;
    for (int a : num)
    {
        if (a<0)
        {
            count++;
        }
    }
    System.out.println("Negative numbers:"
    + count);
}
}
```

Q.15 Explain method overloading with suitable example.

-4 Method overloading is a feature that allows a class to have multiple methods with the same name but different parameter lists.

By providing different method signatures, Java can determine which version of the method to call based on the arguments provided during the method invocation.

eg:
```
class math {
    public int add(int a, int b) {
        return a + b;
    }
    public String add(String str1, String str2){
        return str1 + str2;
    }
}
public class Main {
    public static void main(String args[]) {
        math op = new math();

        int sumInt = op.add(5,10);
        System.out.println("Sum of 5 and 10 (int): "
        + sumInt);
        String result = op.add("Hello", "World");
        System.out.println("Concatenated String: "
        + result);
    }
}
```

Output: Sum of 5 and 10 : 15
Concatenated String: Hello World

Q.16 Create a two dimensional array. Instantiate and initialize it.

—4

```
public class TwoDArray
{
  public static void main (String args[])
  {
    int [][] arr = new int [2][3];
    arr[0][0] = 1;
    arr [0][1] = 2;
    arr [0][2] = 3;

    arr [1][0] = 4;
    arr [1] [1] = 5;
    arr [1] [2] = 6;

    for (int i=0; i<2; i++)
    {
      for (int j=0; j<3; j++)
      {
        System.out.println (arr [i][j] +" ");
      }
      System.out.println ();
    }
  }
}
```

Output :

```
1 2 3
4 5 6
```

Q.17 Differentiate String class and String Buffer class with explanation of its methods.

-> i) String Class

'String' is an immutable class, which means that once a String object is created, its value cannot be changed. When you perform operations on String, it creates new String objects rather than modifying the original one.

- Methods of String class:

① length() : returns no. of characters
② charAt(int index) : returns character at the specified index
③ concat(String str) : concatenates 'str' to the end of original string
④ toLowerCase() : converts into lowercase
⑤ toUpperCase() : converts into uppercase
⑥ trim() : removes leading and trailing white spaces
⑦ indexOf(int ch) : returns the index of the first occurence of the specified character. [-1 if false]
⑧ substring(int beginIndex) : returns a new string that is a substring of the original string, starting from the specified index
⑨ equals(Object obj) : Compares the content of the string with the content of the specified Object

ii) String Buffer Class

> String Buffer is a mutable class, designed for the efficient manipulation of strings. Unlike 'String', you can modify the content of a 'String Buffer' object without creating a new object each time, which makes it more efficient.

- Methods of String Buffer Class:

① append(): appends the specified string at the end.

② insert(): insert the specified string at the specified position.

③ replace(): replaces the substring from 'startIndex' to 'endIndex - 1' with specified string

④ delete(): deletes the characters from 'startIndex' to 'endIndex - 1'

⑤ reverse(): reverses the characters.

⑥ length(): returns the no. of characters.

⑦ charAt(): returns the character at specified index.

⑧ substring(): returns a new string which is a substring, starting from the specified index.

⑨ toString(): converts 'String Buffer' to a 'String'

Q.18 Write a method for computing first n terms of Fibonacci Sequence. Define method main taking value of n as command line argument and calling the method.

→

```
public class Fibonacci
{
   public static void fib(int n)
   {
      int t1=0, t2=1;
      int t3 = t1+t2;
      System.out.print(t1+" "+t2+" ");
      for(int i=2; i<n; i++)
      {
         t3 = t1+t2;
         System.out.print(t3+" ");
         t1 = t2;
         t2 = t3;
      }
      System.out.println();
   }
   public static void main(String args[])
   {
      int n = Integer.parseInt(args[0]);
      fib(n);
   }
}
```

Output: javac fibonacci.java
java fibonacci 6
0 1 1 2 3 5

Q.19 Explain Scanner class with example.

→ Scanner class in Java is a part of the java.util package and it provides a simple way to read input from various sources. It is widely used to read user input and other types of data during program execution.

eg:
```
import java.util.Scanner;
public class Scanner
{
  public static void main (String args[])
  {
    Scanner input = new Scanner (System.in);
    System.out.println ("Enter Your Name: ");
    String name = input.nextLine ();
    System.out.println ("Enter Your Age: ");
    int age = input.nextInt ();
    System.out.println (name + " is " + age
    + " years old.");
    input.close ();
  }
}
```

Output : Enter Your Name : Tony Mikaelson
Enter Your Age : 17
Tony Mikaelson is 17 years old

Q.20 What is wrapper class? What is the use of wrapper class in Java?

→ In Java, a wrapper class is a class that wraps or encapsulates the primitive data types into objects. Java has eight primitive data types (int, float, double, char, byte, short, long, boolean) and each of them has a corresponding wrapper class. They are the part of 'java.lang' package.

- List of Wrapper Class:-
  - 'Integer': Wraps 'int'
  - 'Float': Wraps 'float'
  - 'Double': Wraps 'double'
  - 'Character': Wraps 'char'
  - 'Byte': Wraps 'byte'
  - 'Short': Wraps 'short'
  - 'Long': Wraps 'long'
  - 'Boolean': Wraps 'boolean'

- Use of Wrapper Class:
  1. Collection Framework
  2. Generics
  3. API methods
  4. Parsing and Conversion
  5. Nullability

Q.21 Describe, with examples, the uses of enumeration data types.

→ Enumeration (enum) data type are used to define a fixed set of constants representing a distinct set of values. It provides a more structured and type-safe way to represent a collection of related constants which improves code readability, compile-time safety and better organization.

eg:

```
enum Day {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY,
    SATURDAY, SUNDAY }
public class Enum {
    public state void main (String args[])
    {
        Day today = Day.WEDNESDAY;
        if (today == Day.SATURDAY || today ==
        Day.SUNDAY) {
            System.out.println ("Weekend");
        } else {
            System.out.println ("Weekday");
        }
    }
}
```

Output : Weekday