Project Work

DILBAR YADAV

Q.1 What is object and class? write a program to calculate the area of rectangle using object and class.

→ Object :- An instance of a class containing state (attribute) and behaviour (methods)

Class :- A class is a template or blueprint that describes the behaviour/state that the object of it's type support.

~~The program for calculate the~~
The program of area of rectangle using class and object are given below.

```
Class Rectangle
    int length;
    int breadth;
  int area() {
      return length *
    breadth;
    }
}

Public class main {
    Public static void Main(String [] args) {
            Rectangle rect();
        rect.length = 5;
        rect.breadth = 10;
System.out.println(" Area : " rect.area() );
    }
}
```

Q.2 What is Java programming? Why is Java called platform-Independent?

Java :- A high-level object-oriented programming language which helps us write command (code) to make the computer do a particular task.
Java is one of the manny programming languages we use to instruct the computer.

Platform independent :-
Java programs can run on any device with a JVM (Java virtual Machine), as bytecode is platform-independent.

Q.3 Why d is Java language popular day by day?

Java language popular day by day because its popularity :-
Java is popular due it's platform independence, security features robust API scalability and large community support, making it suitable for a wide range of applications From mobile apps to large enterprise systems.

Q.4 What is looping? How can you construct different types of loops in Java?

Looping :- Looping in programming ~~language~~ concept that allows the repeated execution of a block of code as long as a specified condition is true.

Types of Loops

1. For loop :- Used when the number of interations is known

```
for (int i=0; i<100;
    i++) {
System.out.println (i);
    }
```

2. While loop :- Used when the number of interations is not known and depends on a condition

```
int i=0;
while (i<10) {
System.out.println (i);
    i++;
    }
```

3. Do-while loop :- Do-while loop is similar to the while-loop but the block of code executes at least once, regardless of the condition.

```
int i=0;
do {
System.out.println (i);
    i++;
} while (i<10);
```

#Q.5 What is Operator? Explain different types of Operators used in Java.

→ Operator :- Operator are special symbol that perform specific operations operations on operands (variable or value) Java provides a rich set of operators to manipulate data.

The different types of Operator used in Java is given below.

* Arithmetic operators :- These operator perform basic Mathmatical Operator.
  - (+) :- Addition
  - (-) :- Subtraction
  - (*) :- Multiplication
  - (/) :- Division
  - (%) :- Modulus (remainder after division)

* Assignment Operators :- These Operator are used to design values to Varibles.
  - (=) :- Assigns a value to a Variable.
  - (+=), (-=), (*=), (%=) :- Compound assignment Operators that combine arithmetic operator with assignment

* Unary Operators :- These Operators operate a single Oprend
  - (+) :- Unary plus (indicates Positive value)
  - (-) :- Unary Minus (indicated Negative value)
  - (++) :- Increment (increases the value by 1)
  - (--) :- Decrement (decreases the value by 1)

* Conditional (Tennary) Operator :- A Shorthand For the if-else statement
  - 'conditional ? value-if-true : value-if-false'

* Logical Operators:- These operators are used to combine and manipurate boolean values.
  - (&&):- Logical AND
  - (||):- Logical OR
  - (!):- Logical NOT

* Bitwise Operators:- These Operator perform operations at the bit level
  - (&):- Bitwise AND
  - (|):- Bitwise OR
  - (^):- Bitwise XOR (exclusive OR)
  - (~):- Bitwise NOT
  - (<<):- Left shift
  - (>>):- Right shift

* Comparison Operator:- These Operator Compare two values and return a boolean result ('True' or 'False')
  - (==):- Equal to

Q6 * What is array? Write a program to store and display 50 numbers using array.

Array:- An array is a collection of elements of the same type. Stored in continous memory locations.
It allows you to store and access multiple values using a single variable name and an index.

A program to store and display 50 numbers using array. is given below.

```
Pubic Class Dilbar {
  Public Static void Main (String [] args) {
  int [] numbers = new
    int [50];
                  // store number in array
for (int i=0; i<50; i++) {
    numbers [i] = i + 1;
                }
// Display numbers
for(int i=0; i<50; i++) {
System.out.println (" Numbers at index " + i + " is: " +
                  numbers [i]);
                }
              }
            }
```

Q.8 What is inheritance? explain its types with examples.

Inheritance :- In Java is a core concept of object-oriented Programming (OOP) mining Mechanism in where a new class (child class) inherits ~~in OOP~~ the properties and behaviours of an existing class (Parent class). This promote code reusability.

Types of inheritance with examples are given below.

* Single Inheritance :- one child class inherits from one Parent class.

Eg
```
Class Animal { void
    eat()
  {
System.OUT.println ("Eating...");
        }}
Class Dog extends Animal
   {
    void bark()
{   system.OUT.println("Barking...");
        }}
```

* Multilevel Inheritance :-

A child class inherits from a parent class, which inherits from another cla

Eg

~~Class Animal~~

```
Class Animal {void eat()
{ System.OUT.println("Eating....");
        }}
```

```
Class Dog extends Animal
   { void bark()
  { system.OUT.println("Barking...");
        }}
```

```
Class PuBBy extends Dog
   { void weep()
{
 System.OUT.println("Weeping...");
        }}
```

* Hierachial Inheritance:-
A Multiple child classes inherit From the Same Parent class.

E.g
```
Class Animal{
    void eat(){
System.out.println("Eating...");
    }
}

Class dog extends
Animal{
    void bark(){
System.out.println("Barking...");
    }
}

Class cat extends
Animal{
    void meow(){
System.out.println("Meowing...");
    }
```

Q.9# What is Polymorphism? Explain with accurate examples(overriding)

→ Polymorphism:- Polymorphism is a concept in object-oriented programming that allows methods or object to behave differently based on their data types or Class.
There are two main types of Polymorphism
* Compile-time Polymorphism (method ~~overriding~~ overloading)
* Run-time Polymorphism (method ~~overriding~~ ~~overloading~~)

* Compile time Polymorphism (method overloading):-

→ Compile time Polymorphism is also known as method overloading. It occur when different methods have same same but different parameters within the same class. The appropriate method to execute is determined by the compiler based on the method signature.

* Runtime Polymorphism:-

Runtime Polymorphism is also known as method overriding. It occurs when a subclass provides a specific implementation for a method that is already defined in its superclass. The method to execute is determined during runtime based on the actual object refered to.

*Q.10 What is OOP? explain different principle of OOPs.

→ OOP is stand for Object-oriented Programming systems, which is a programming paradigm based on the concept of "object" that can contain data and code. The data stored in fields and the code is stored in procedures.

Principles of OOPs are given below.

- Encapsulation:- is the principle of hiding the internal details or, mechanics of how an object does something. The object provides methods that allows the user to interact with it without needing to understand its internal implemention. This reduces complexity and increases reusability

* Abstraction:- Abstraction Simplifies Complex system by Modeling Classes that represent essential Features without Including background details or explanation. It allows programmer to work with higher-level concepts, while hiding the underlying implementation, which makes the system easier to work with modify.

* Inheritance:- Inheritance is a machanism where a new class (child class) derives properties and behavior (methods) from an existing class (parent class). This promote code reusability and establishes a natural relationship between classes, reducing redundancy in code.

* Polymorphism:- Polymorphism means "Manny Forms," and it allows methods to do different things based on the object it is acting upon even though they share the same name. This can be implemented through method overriding or method overloading enabling Flexibility in Programming.

Q.11 What is Constructor? Explain different types of Constructor with eg

→ A Constructor is a special method in Object-oriented Programming that is automatically called when an object of a class is created. Its main purpose is to initialize the object's attributes and allocate resources. Constructor have same ~~class~~ name as the class and do not have a return type.

The different types of Constructor with examples are given below

* ~~De~~ Default Constructor
* Parameterized Constructor
* Copy Constructor.

1* Default Constructor:-

A default constructor that does not take any parameters. It initiizes object attributes to default values.

Example,

```
Class Rectangle {
  Pubic:
    int width, height;
  // Default Constructor
  Rectangle () {
    width = 0;
    height = 0;
  }
};
int main () {
  Rectangle rect; // Default Constructor is called
  return 0;
}
```

* Parameterized Constructor:-

A Parameterized Constructor that takes arguments to initiaiize object attributes with specific value.

Examples:

```
Pubic Class Rectangle {
  int width, height;
  // Parameterized Constructor
  Rectangle (int w, int h) {
    width = w;
    height = h;
  };
int main () {
  Rectangle rect (10, 20); // Parameterized constructor is called
  return 0;
}
```

* Copy constructor:- A copy constructor that intializes an object using another object of the same class. It is used to copy data from one object to another.

Example;

```
Public class Rectangle {
    int width, height;
    // Parameterized Constructor
    Rectangle(int w, int h) {
        width = w;
        height = h;
    }
    // copy constructor
    Rectangle(const Rectangle &rect) {
        width = rect.width;
        height = rect.height;
    }
};
int main() {
    Rectangle rect1(10, 20); Parameterized Constructor
    Rectangle rect2(rect1); //
    return 0;
}
```

#Q2 What is abstract class? Different between class and abstract class.

Abstract:- 'Abstract' in Java is a keyboard used to define abstract classes and abstract methods. An abstract class cannot be instantiated directly and servers as a blueprint for subclasses. It can have abstract method or non-abstract methods. Abstract method are declared without implementations and must be overridden by subclasses.

Different between class and Abstract class are given below.

	Class	Abstract Class
*	Class can create objects instanciated	Abstract class can't create objects instanciated
*	All methods must have complete code (method bodies)	Abstract class can have both abstract methods (no code) and concrete methods (with code)
*	Class used to define the behaviour of objects	* Abstract class used as a base class to be extended by other classes.
*	Class cannot be declared as abstract.	* must be declared with the abstract keyword.
*	Class have a constructor used when creating objects	* Abstract class have a constructor, but it is used by subclasses.
*	Class allows direct of objects using the keyword	* objects are ~~creation of objects~~ created only through subclasses

#Q13 What is interface? What are the uses of interface? Explain with examples.

→ Interface :- In Java, an interface is a collection of instructions for methods that a class is required to implement. It specifies the methods that must be present in any class that implements the interfaces, defining a contract that class must abide by. Java interface offer a means of achieving multiple inheritance-like behaviour and allow programs to interact with other classes using a shared set of method.

The uses of interface are given below.

*1 Defining a contact :- Specifies methods that implementing classes must provide, ensuring consistent behaviour across different classes.

Eg:-

```
Interface Animal{
    void makesound();
}
```

*2 Supporting multiple Inheritance :- A class can Implement multiple Interfaces, overcoming Java's limitations of single Inheritance with classes.

Example:-

```
Interface Flyable{
    void fly();
}
Interface swimmable{
    void swim();
}
```

*3 Providing default methods :- Since Java 8, interfaces can include default methods with an implementation allowing new methods to be added without breaking existing classes.

Example.

```
Interface printer{
    void print();
    default void scan()
    { System.out.println("scanning"); }
}
```

The uses of interface are given below.

*1 Defining a contact :- Specifies methods that implementing classes must provide, ensuring consistent behaviour across different classes.

Eg:-
```
Interface Animal{
    void makesound();
}
```

*2 Supporting multiple Inheritance :- A class can implement multiple interfaces, overcoming Java's limitations of single Inheritance with classes.

Examples:-
```
Interface Flyable{
    void fly();
}
Interface Swimmable{
    void swim();
}
```

*3 Providing default methods :- Since Java 8, interfaces can include default methods with an implementation allowing new methods to be added without breaking existing classes.

Example.
```
Interface Printer{
    void print();
    default void scan()
    { System.out.println("Scanning"); }
}
```

Q.14 What is package? How Can you implement package?

→ Package:- A set OF Classes and Interfaces grouped together are known as Package. or package is a way to group releted classes and interfaces helping origanize code, Prevent naming conflicts, and improve project managment is called Package.

*1 Create a package:- At the top OF Java File, Use OF Package keyword followed by the package name.
e.g package mypackage;

*2 Save the File:- Save the File in a directory structure that matches the Package name (e.g:- a Folder named mypackage).

*3 Use the Package:- To Use a class From the package, Use the Import Statement in Other classes.
e.g:- Import mypackage.MyClass

*4 Compile the Java File:- Use the Javac compiler to compile the Java File. The compiler will store the compiled. Class File in the Corresponding Package Folder.

#Q15 What is exeception? How does exeption differ from error?

→ Exeception:- Exception is an event, which occurs during the execution OF a Program that disrupts the normal Flow OF the Program's instruction is called exception.
There are manny typical causes For exeption is.
Some are given below.
* ~~Incorrect data~~ Invalid Input data
* Loss OF network Connectivity
* Request For missing or non-existent Files.

Exception differs from error

Exception	Error
* An event that disrupts normal program flow and can often be handled	* A serious problem that represents a failure beyond the program's control
* Subclasses of the Exception class	* Subclasses of the Error class
* Can be caught and handled using try-catch blocks	* Generally cannot be caught or handled by the program.
* They are defined in Java language exception package	* They are defined in Java language Error package.
* Exception include both checked as well as unchecked type	* All errors in Java are unchecked type.
* Ex Unchecked exception occurs in runtime whereas checked exceptions occur at compile time	* Error can occur at compile time.
* Eg:- Checked exceptions, Unchecked exceptions, Array index, Arithmetic exception, NULL Pointer, SQLE etc.	* Eg:- Java.lang.stackoverflow error, Java.lang.Out of memory Error.

*Q16 What is thread? explain different steps of thread life cycle.

→ Thread:- Thread is the direction or path that is taken while a program is being executed. Generally all the programs have at least one thread, known as the main thread that is provided by the JVM at the starting of the program's execution. It enables multitasking and efficient use of CPU resources. Threads can be created by extending the thread class or implementing the Runnable interface.

Explaination of different steps of thread life cycle are given below.

*1 New (or Born):- The thread is created but not yet started. It is in New state.
e.g Thread t = new thread ();

*2 Runnable:- The thread is ready to run and waiting for CPU time. It moves to the RUNNABLE state after calling start().
e.g t.start();

*3 Blocked:- The thread is waiting for resources or locks that are currently unavailable. It enters the BLOCKED state while waiting to acquire a monitor lock.

*4 Waiting:- The thread is waiting for another to perform a particular action (e.g. waiting indefinitely). It enters the WAITING state when it calls.
e.g synchronized (objects){
objects.wait(); }

*5 Timed Waiting:- The thread is waiting for specific period. It enters the TIMED_WAITING state when calling Thread.sleep() with a specified time.

*6 Terminated:- The thread is has completed its execution or terminated due to an exception it is in the TERMINATED state.

Q.17 Differentiate between Final & Finally?

→	Final	Finally
*	Final is a keyword	* Finally is a block
*	Method - cannot overridden	* method can overridden
*	Final: To declare constants, Prevent method overriding, or Prevent class inheritance.	* Finally: To ensure code runs after a try block, regardless of whether an exception occurs.
*	Final with Variables, method, or classes	* Finally with try and catch blocks.
*	Final makes variables constant, methods non-overrictable, and classes non-extendable	* Finally guarantees a block of code executes after try/catch.
*	E.g of Final: final int MAX = 10;	E.g of Finally: Try { // code } finally { // cleanup }

Q.18 Differentiate between method overloading and overriding.

→	Method overloading	Method overriding
*	Method overloading is a compile-time Polymorphism.	* Method overriding is a run-time Polymorphism.
*	It helps to increase the readability of the program.	* It is used to grant the specific implementation of the method which is already provided by its present class or superclass.
*	It occurs within the class	* It is performed in two classes with inheritance relationship
*	Method overloading may or may not require in inheritance	* Method overriding always needs Inheritance.
*	Static binding is beings used for overloaded methods	* Dynamic binding is being used for overriding methods.

P.T.O

* In method Overloading, the Methods Must have the same Name and different signature	* In method overriding, Must have the same name and same signature.
* In method overloading, the return type can or cannot be the same, but we have just have to change the parameter	* In method overriding the return type must be the same ~~name~~ and or co-variant

#Q19 Differentiate between OOP and POP

→ Differentiate between OOP and POP are given below.

OOP	POP
* OOP is stand For Object oriented programming	* POP is stand for Procedural oriented programming
* The main focous of OOP is on the data	* The main Focous of POP is on the Procedures.
* It has access specifiers such as Pubilc, Priite etc	* In this Functions use global data
* It Provides data hiding, data associated with the Program. so Security provide	* POP does not Provide any data Security.
* Ease Of Modification	* Modification is difficult.
* It follows bottom up approach	* It follows top-down approach.
* programs are divided into parts known as Object	* In this program are divided in to Functions
* Examples:- Java, Perl	* Examples:- C, COBOL etc.

#Q.20 Explain this Super Keyword with appropriate examples.

→ Explaination of super keyword with appropriate examples are given below.

The Superkeyword in python is used to call methods from parent class from within a child class. It allows you to access inherited methods that haven been overhidden in a child class. It is commonly used to:-

1 Call the Parent class constructor :- This ensures that the parent class is properly initialized when a child class is created

2. Access overhidden methods :- It allows a child class to call methods of the parent class that have been overhidden in the child class.

* Examples:- Using super to call Parent class constructor.

```
Class Animal:
def __init__(self, name):
    self.name = name
class Dog(Animal):
def __init__(self, name, bread):
super().__init__(name) #
* Calls the Parent class constructor    self.bread = bread

# Create an instance of Dog
dog = Dog("Buddy", "Golden Retriever")
print(dog.name)
# OUTPUT: Buddy.
```