JAVA QB UNIT-1

1. What is Java? List and explain different features of Java.

Ans-Java is an object-oriented programming language and a platform. Java was developed by Sun Microsystems (which is now the subsidiary of Oracle) in the year 1995. James Gosling is known as the father of Java.

Features of java:-

1. Simple-Java is very easy to learn, and its syntax is simple, clean and easy to understand.
2. Object-Oriented-Java is an object-oriented programming language. Everything in Java is an object.
3. Portable-Java is portable because it facilitates you to carry the Java bytecode to any platform.
4. Secured-Java is best known for its security. With Java, we can develop virus-free systems.

2. Explain Java tokens.

Ans-The Java compiler breaks the line of code into text (words) is called Java tokens. These are the smallest element of the Java program. The Java compiler identified these words as tokens.

Keywords, Identifiers, Literals, Operators, Separators, Comments are all java tokens.

3. What is data type? Explain different data types available in Java.

Ans-Data types specify the different sizes and values that can be stored in the variable.

Data types are divided into two groups:

- Primitive data types -A primitive data type specifies the size and type of variable values, and it has no additional methods.

  byte, short, int, long, float, double, boolean and char.

- Non-primitive data types – Non-primitive data types are called reference types because they refer to objects.

  String, Arrays and Classes.

4. What is variable?

Ans-Variables are containers for storing data values. In Java, there are different types of variables, for example: String - stores text, such as "Hello". String values are surrounded by double quotes. int - stores integers (whole numbers), without decimals, such as 123 or -123.

5. Explain different operators available in Java.

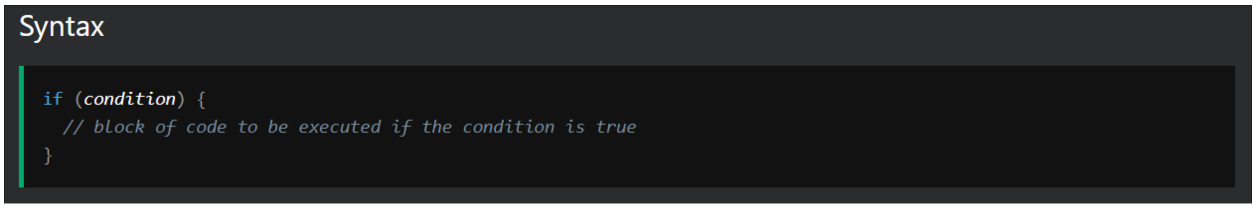
Ans-Operators are used to perform operations on variables and values.

Java divides the operators into the following groups:

- Arithmetic operators-Arithmetic operators are used to perform common mathematical operations.
- Assignment operators-Assignment operators are used to assign values to variables.
- Comparison operators-Comparison operators are used to compare two values.
- Logical operators-Logical operators are used to determine the logic between variables or values.

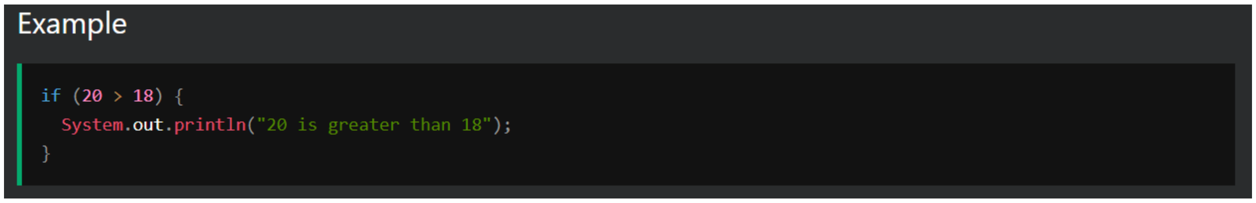
6. Explain if condition with suitable example.

Ans-The if statement is used to specify a block of Java code to be executed if a condition is true.



Syntax

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

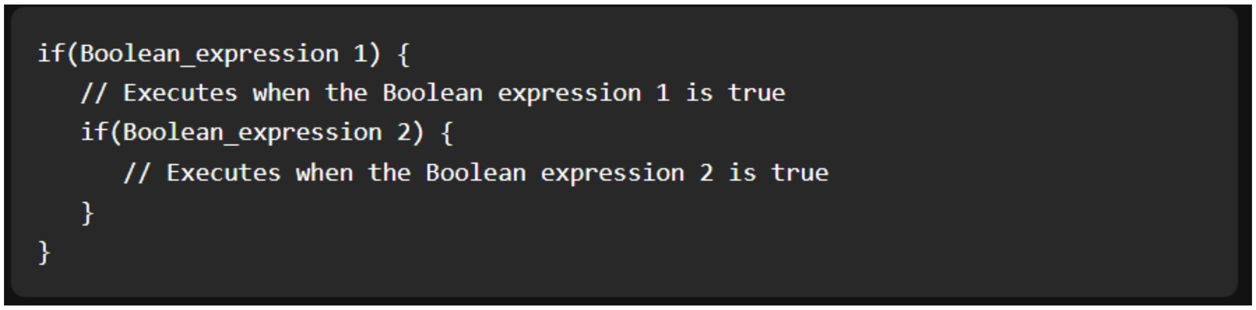


Example

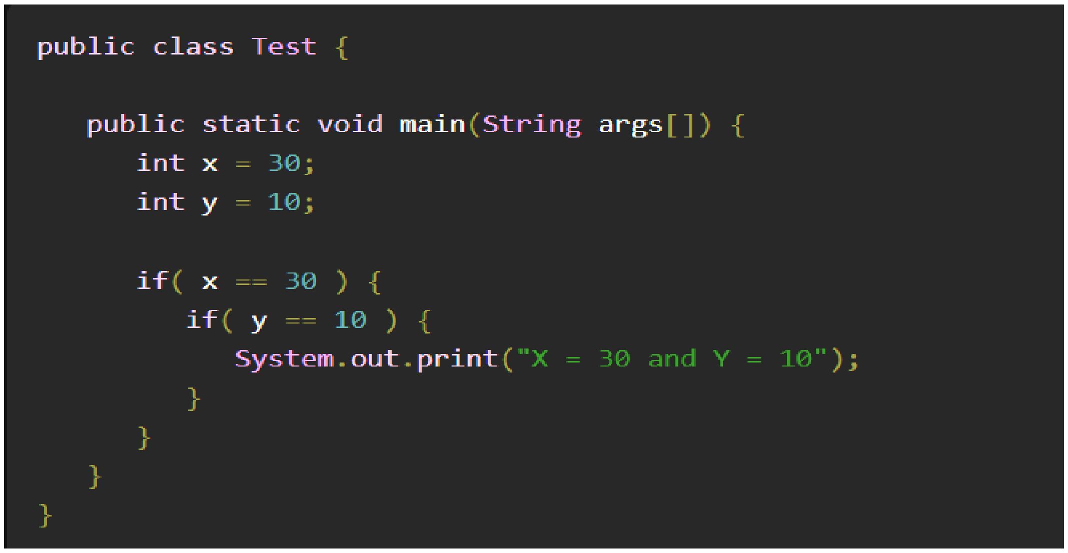
```
if (20 > 18) {
  System.out.println("20 is greater than 18");
}
```

7. What is nested if condition? Explain it with suitable example.

Ans-In Java, it is also possible to use if..else statements inside an if...else statement. It's called the nested if...else statement.



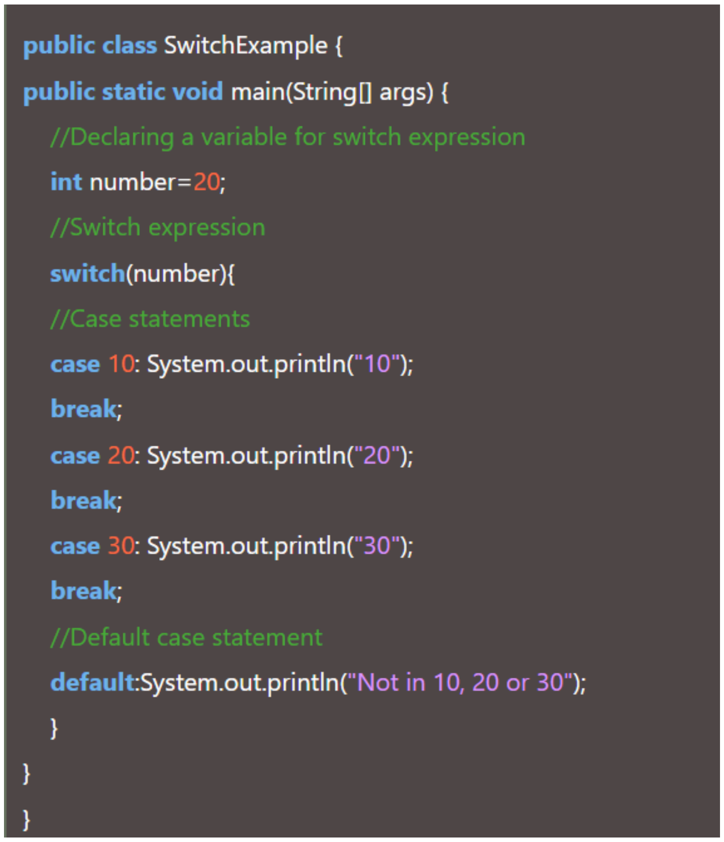
```
if(Boolean_expression 1) {
   // Executes when the Boolean expression 1 is true
   if(Boolean_expression 2) {
      // Executes when the Boolean expression 2 is true
   }
}
```



```java
public class Test {

   public static void main(String args[]) {
      int x = 30;
      int y = 10;

      if( x == 30 ) {
         if( y == 10 ) {
            System.out.print("X = 30 and Y = 10");
         }
      }
   }
}
```

8. Explain switch case with suitable example.

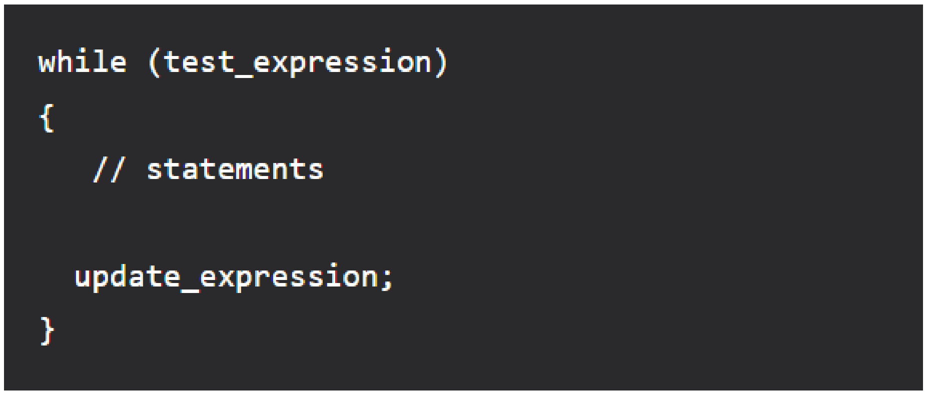
Ans-Switch case statement evaluates a given expression and based on the evaluated value(matching a certain condition), it executes the statements associated with it.



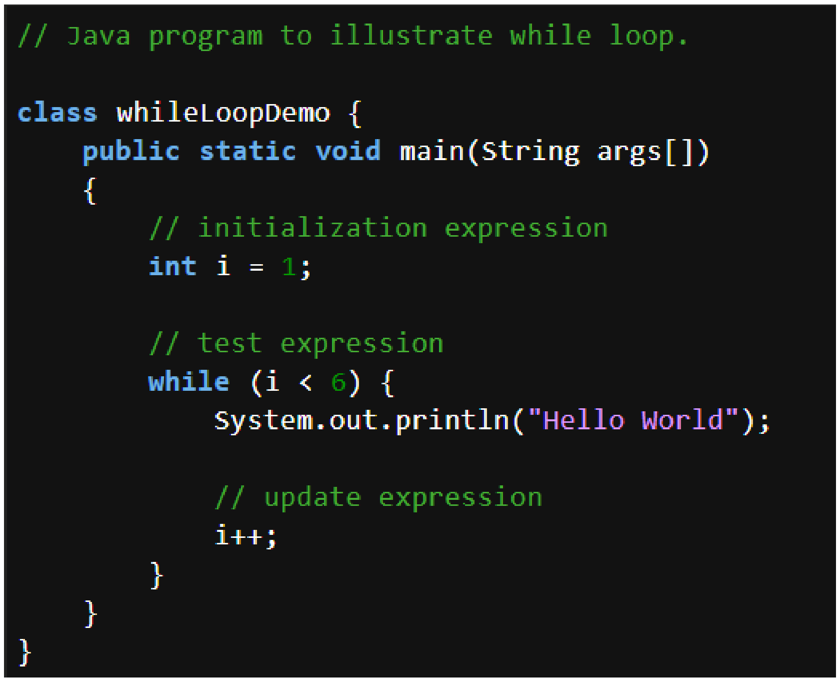
```java
public class SwitchExample {
public static void main(String[] args) {
   //Declaring a variable for switch expression
   int number=20;
   //Switch expression
   switch(number){
   //Case statements
   case 10: System.out.println("10");
   break;
   case 20: System.out.println("20");
   break;
   case 30: System.out.println("30");
   break;
   //Default case statement
   default:System.out.println("Not in 10, 20 or 30");
   }
}
}
```

9. Explain while loop with suitable example.

Ans-While loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition.



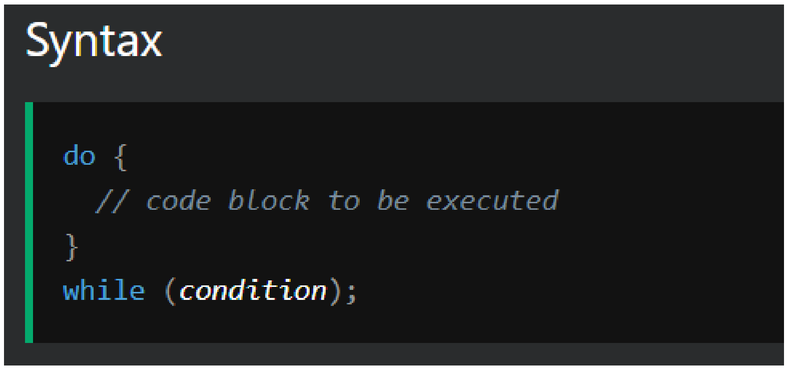
```
while (test_expression)
{
   // statements

  update_expression;
}
```



```
// Java program to illustrate while loop.

class whileLoopDemo {
    public static void main(String args[])
    {
        // initialization expression
        int i = 1;

        // test expression
        while (i < 6) {
            System.out.println("Hello World");

            // update expression
            i++;
        }
    }
}
```

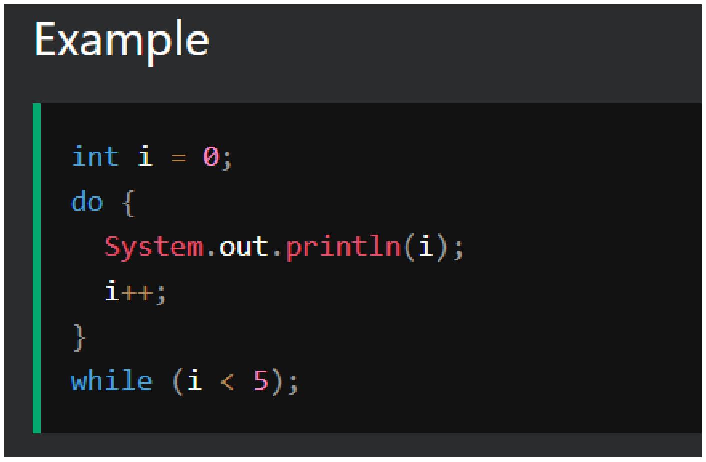
10. Explain do while loop with suitable example.

Ans-The do/while loop is a variant of the while loop.



Syntax

```
do {
  // code block to be executed
}
while (condition);
```

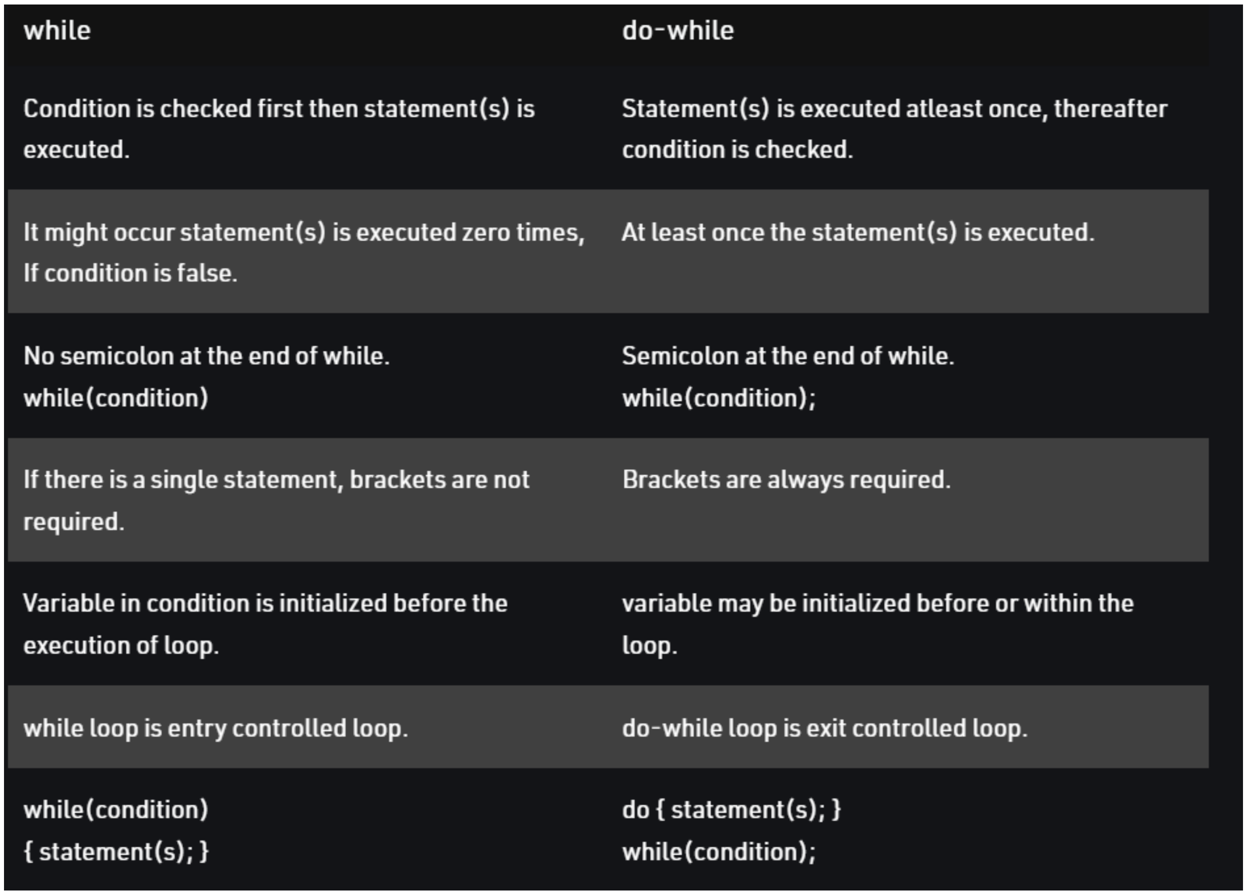


Example

```
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

11. Differentiate between while and do while loop.

Ans-



while	do-while
Condition is checked first then statement(s) is executed.	Statement(s) is executed atleast once, thereafter condition is checked.
It might occur statement(s) is executed zero times, If condition is false.	At least once the statement(s) is executed.
No semicolon at the end of while. while(condition)	Semicolon at the end of while. while(condition);
If there is a single statement, brackets are not required.	Brackets are always required.
Variable in condition is initialized before the execution of loop.	variable may be initialized before or within the loop.
while loop is entry controlled loop.	do-while loop is exit controlled loop.

while(condition)
{ statement(s); }
	do { statement(s); }
	while(condition);

12. What is typecasting? Explain it's type with example.

Ans-Converting one datatype into another is known as type casting or, type-conversion.

In Java, there are two types of casting:

- Widening Casting (automatically) - converting a smaller type to a larger type size

byte -> short -> char -> int -> long -> float -> double

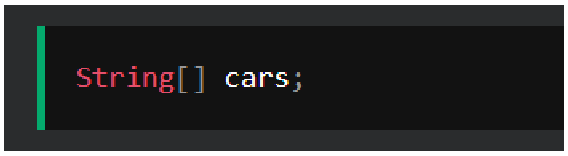
- Narrowing Casting (manually) - converting a larger type to a smaller size type

double -> float -> long -> int -> char -> short -> byte

13. What is array? How to declare array in java.

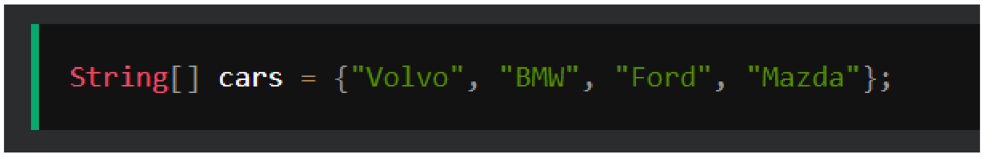
Ans-An array in Java is a group of like-typed variables referred to by a common name.

To declare an array, define the variable type with square brackets:



```
String[] cars;
```

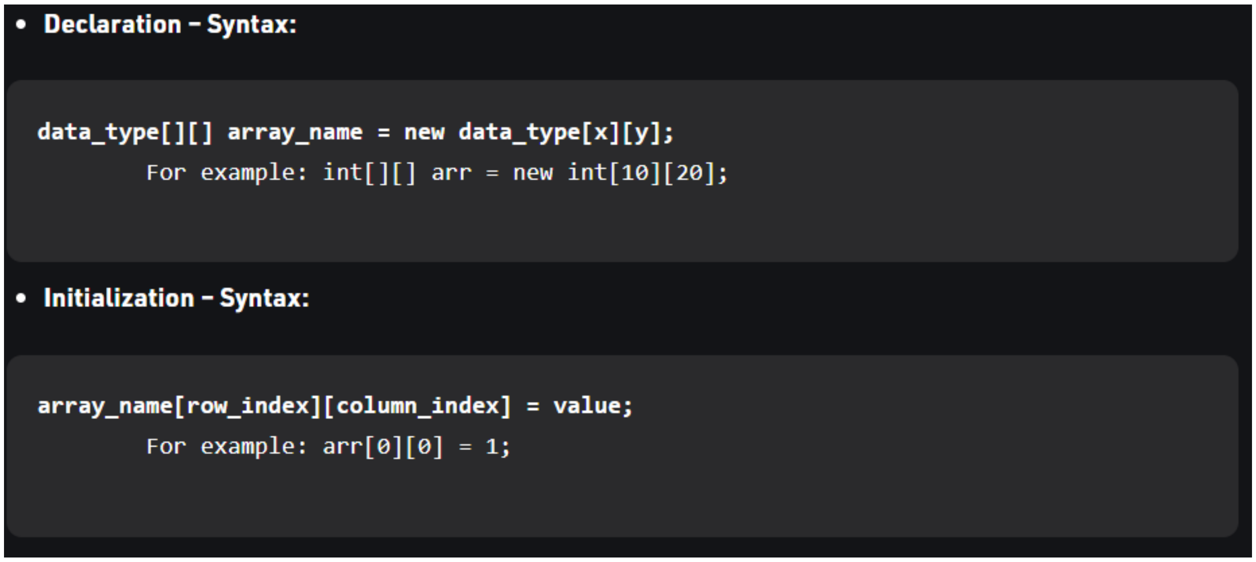
To insert values to it, you can place the values in a comma-separated list, inside curly braces:



```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

14. Explain concept of 2D array in java.

Ans-Two dimensional array is the simplest form of a multidimensional array. A two – dimensional array can be seen as an array of one – dimensional array for easier understanding.



- **Declaration – Syntax:**

```
data_type[][] array_name = new data_type[x][y];
        For example: int[][] arr = new int[10][20];
```

- **Initialization – Syntax:**

```
array_name[row_index][column_index] = value;
        For example: arr[0][0] = 1;
```

15. List and explain different features of OOPs.

Ans-

1. Classes and Objects-A Class is like an object constructor, or a "blueprint" for creating objects.
2. Methods-A method is a block of code which only runs when it is called.
3. Constructors-A constructor in Java is a special method that is used to initialize objects.
4. Encapsulation-The meaning of Encapsulation, is to make sure that "sensitive" data is hidden from users.

5. Inheritance- It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class.
6. Abstraction-Data abstraction is the process of hiding certain details and showing only essential information to the user.
7. Polymorphism-Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

16. Define following term:

e. Overloading -Method Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters, or a mixture of both.

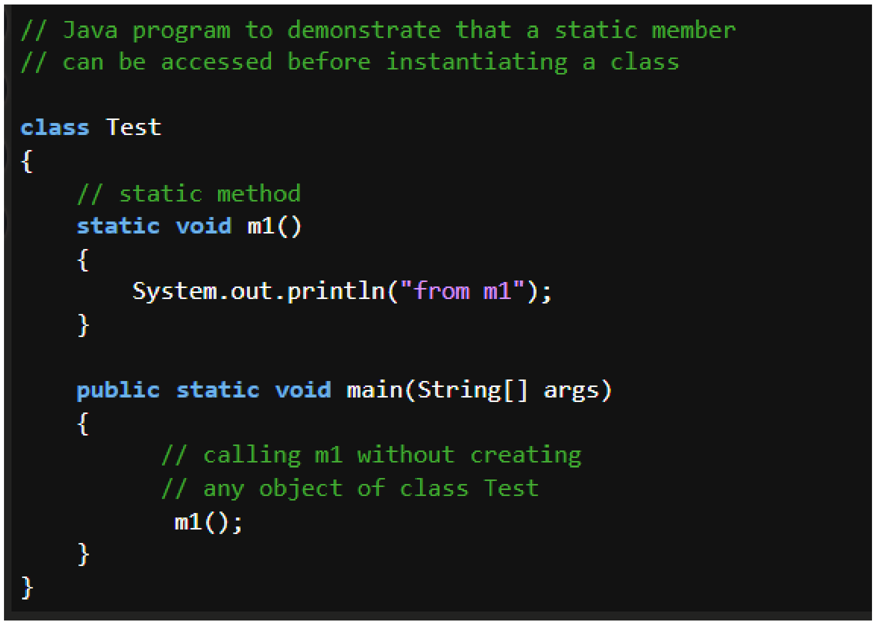
f. Overriding-Method overriding occurs when a subclass provides a particular implementation of a method declared by one of its parent classes.

i. Abstract class-Abstract class is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

k. Interface-An Interface in Java programming language is defined as an abstract type used to specify the behavior of a class.

17. Explain static keyword with suitable example.

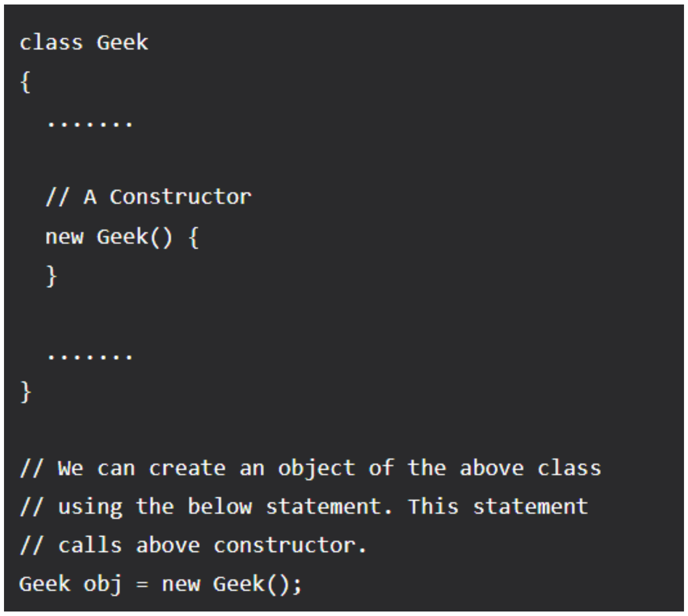
Ans-The static keyword in Java is mainly used for memory management. The static keyword in Java is used to share the same variable or method of a given class. The users can apply static keywords with variables, methods, blocks, and nested classes. The static keyword belongs to the class than an instance of the class. The static keyword is used for a constant variable or a method that is the same for every instance of a class.



```java
// Java program to demonstrate that a static member
// can be accessed before instantiating a class

class Test
{
    // static method
    static void m1()
    {
        System.out.println("from m1");
    }

    public static void main(String[] args)
    {
          // calling m1 without creating
          // any object of class Test
           m1();
    }
}
```

18. Explain purpose of constructor with suitable example.

Ans-A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes.



```
class Geek
{
  .......

  // A Constructor
  new Geek() {
  }

  .......
}

// We can create an object of the above class
// using the below statement. This statement
// calls above constructor.
Geek obj = new Geek();
```

19. What is constructor? Explain it's type.

Ans- A constructor in Java is a special method that is used to initialize objects.

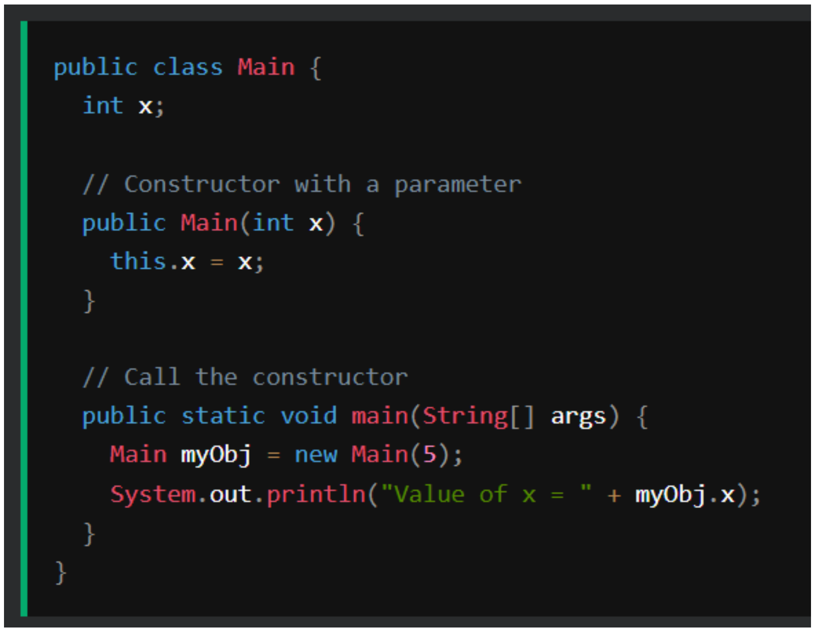
Constructor(s) of a class must have the same name as the class name in which it resides.

- Default Constructor: A constructor that has no parameter is known as the default constructor.
- Parameterized Constructor: A constructor that has parameters is known as parameterized constructor.

20. Explain purpose of this keyword with suitable example.

Ans- The this keyword refers to the current object in a method or constructor.

The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name.



```java
public class Main {
  int x;

  // Constructor with a parameter
  public Main(int x) {
    this.x = x;
  }

  // Call the constructor
  public static void main(String[] args) {
    Main myObj = new Main(5);
    System.out.println("Value of x = " + myObj.x);
  }
}
```

21. What is Inheritance? List it's type.

Ans- It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class.

Important Terminologies

- Super Class: The class whose features are inherited is known as a superclass(or a base class or a parent class).
- Sub Class: The class that inherits the other class is known as a subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.

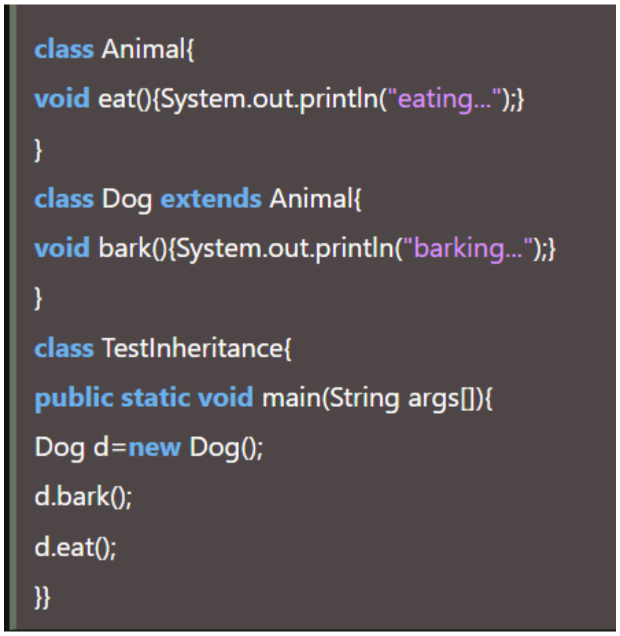
Types of Inheritance:

1. Single Inheritance: In single inheritance, subclasses inherit the features of one superclass.
2. Multilevel Inheritance: In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class.
3. Hierarchical Inheritance: In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass.

4. Multiple Inheritance (Through Interfaces): In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes.
5. Hybrid Inheritance:(Through Interfaces): It is a mix of two or more of the above types of inheritance.

22. Explain single Inheritance with suitable example.

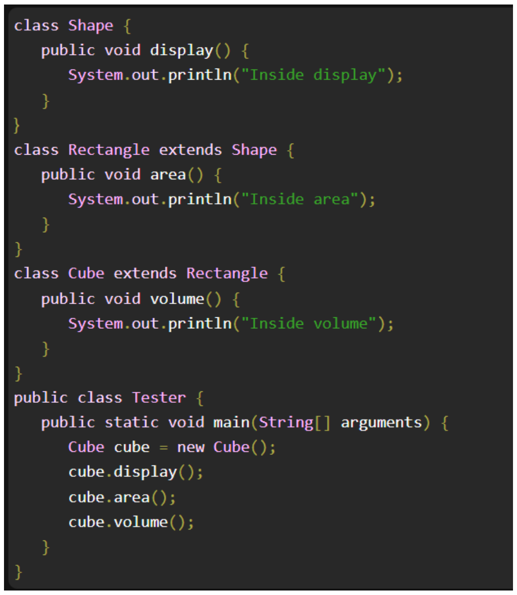
Ans-In single inheritance, subclasses inherit the features of one superclass.



```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

23. Explain multilevel Inheritance with suitable example.

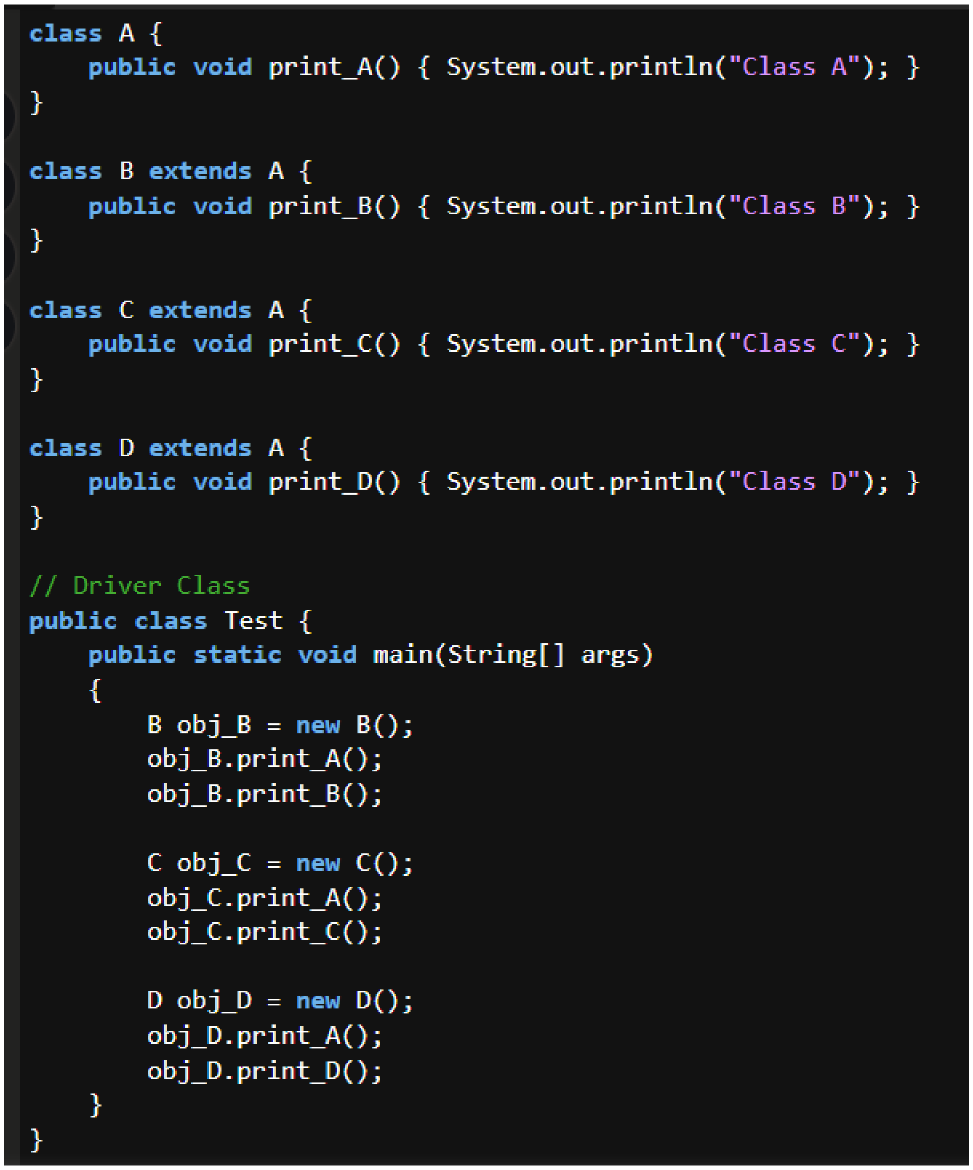
Ans- In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class.



```
class Shape {
   public void display() {
      System.out.println("Inside display");
   }
}
class Rectangle extends Shape {
   public void area() {
      System.out.println("Inside area");
   }
}
class Cube extends Rectangle {
   public void volume() {
      System.out.println("Inside volume");
   }
}
public class Tester {
   public static void main(String[] arguments) {
      Cube cube = new Cube();
      cube.display();
      cube.area();
      cube.volume();
   }
}
```

24. Explain hierarchical Inheritance with suitable example.

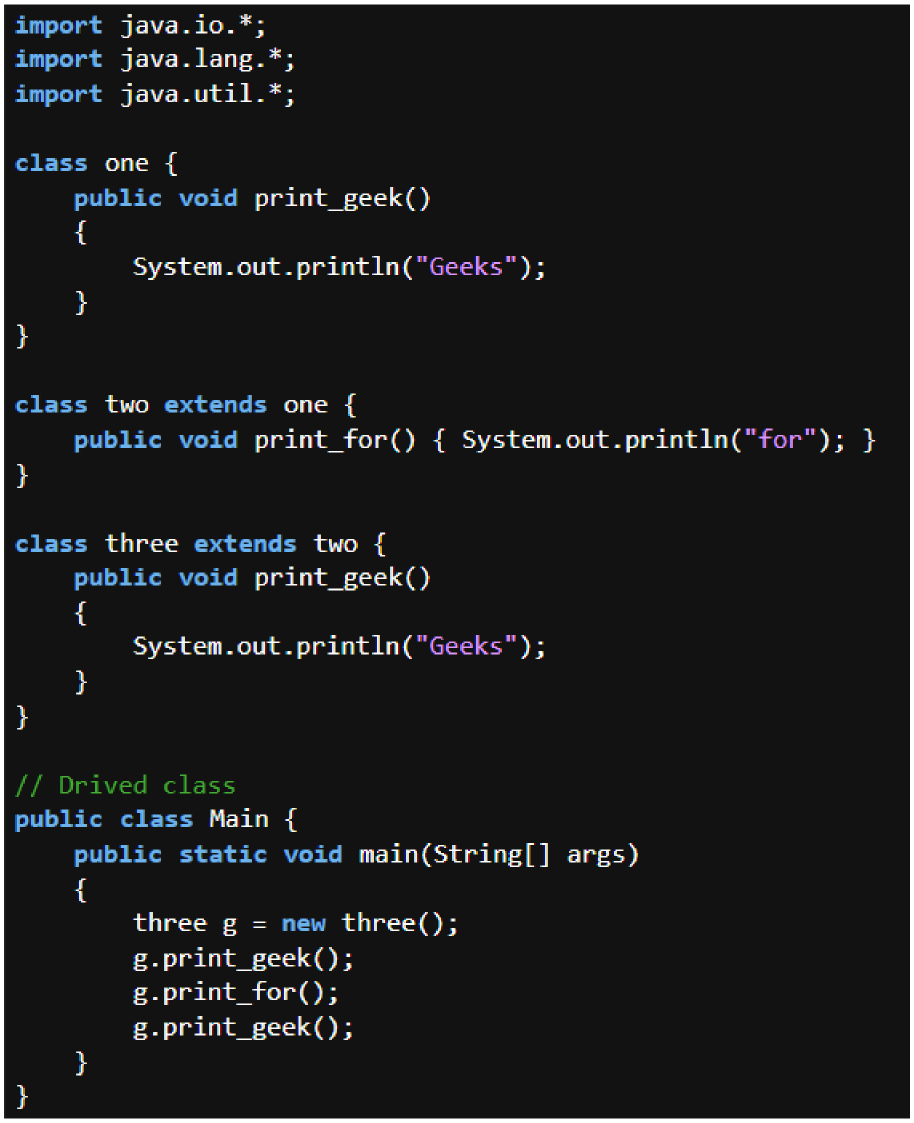
Ans- In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass.



```java
class A {
    public void print_A() { System.out.println("Class A"); }
}

class B extends A {
    public void print_B() { System.out.println("Class B"); }
}

class C extends A {
    public void print_C() { System.out.println("Class C"); }
}

class D extends A {
    public void print_D() { System.out.println("Class D"); }
}

// Driver Class
public class Test {
    public static void main(String[] args)
    {
        B obj_B = new B();
        obj_B.print_A();
        obj_B.print_B();

        C obj_C = new C();
        obj_C.print_A();
        obj_C.print_C();

        D obj_D = new D();
        obj_D.print_A();
        obj_D.print_D();
    }
}
```

25. How multiple Inheritance is achieved in java explain with suitable example.

Ans- In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes.



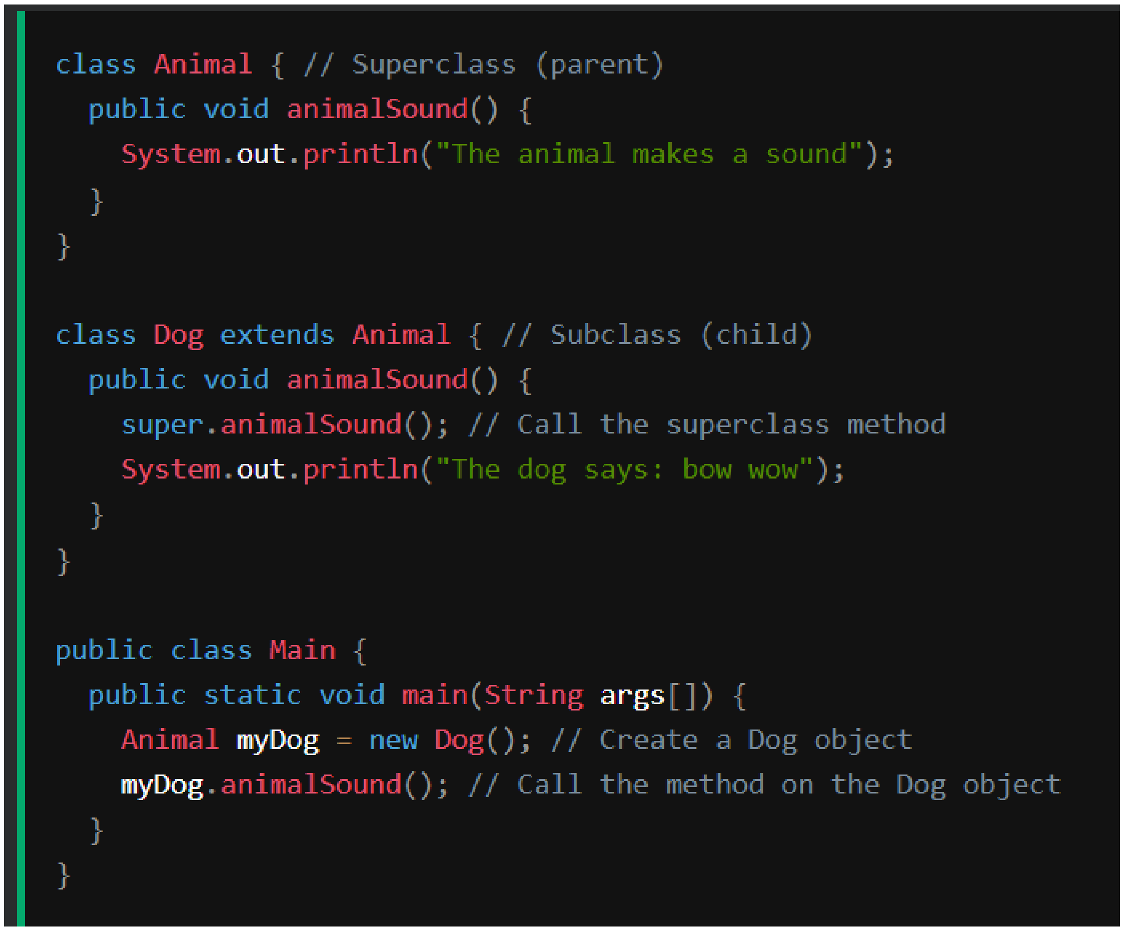
```java
import java.io.*;
import java.lang.*;
import java.util.*;

class one {
    public void print_geek()
    {
        System.out.println("Geeks");
    }
}

class two extends one {
    public void print_for() { System.out.println("for"); }
}

class three extends two {
    public void print_geek()
    {
        System.out.println("Geeks");
    }
}

// Drived class
public class Main {
    public static void main(String[] args)
    {
        three g = new three();
        g.print_geek();
        g.print_for();
        g.print_geek();
    }
}
```

26. Explain purpose of super keyword.

Ans- The super keyword refers to superclass (parent) objects.

It is used to call superclass methods, and to access the superclass constructor.

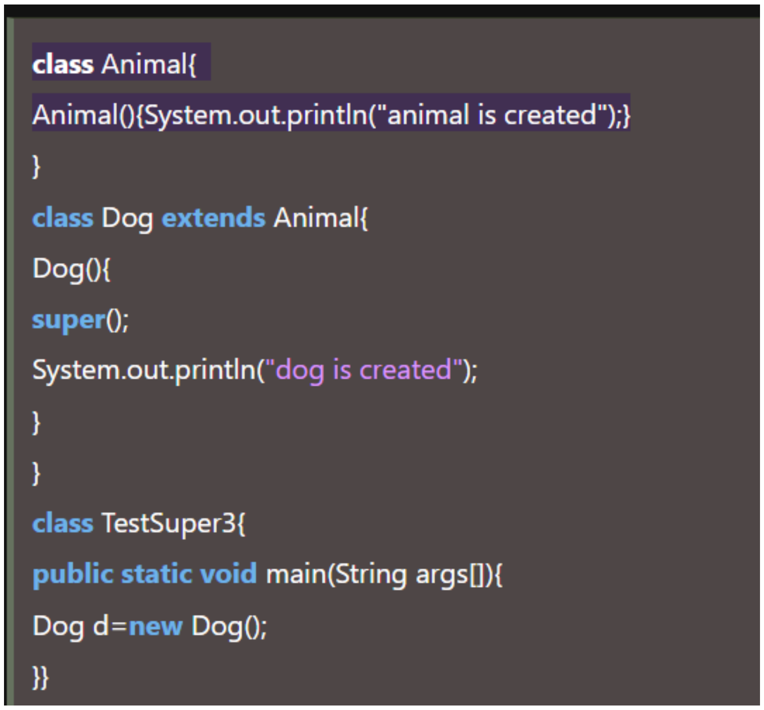
The most common use of the super keyword is to eliminate the confusion between superclasses and subclasses that have methods with the same name.



```java
class Animal { // Superclass (parent)
  public void animalSound() {
    System.out.println("The animal makes a sound");
  }
}

class Dog extends Animal { // Subclass (child)
  public void animalSound() {
    super.animalSound(); // Call the superclass method
    System.out.println("The dog says: bow wow");
  }
}

public class Main {
  public static void main(String args[]) {
    Animal myDog = new Dog(); // Create a Dog object
    myDog.animalSound(); // Call the method on the Dog object
  }
}
```

27. How super keyword is used to call constructor explain with example.

Ans- The super keyword can also be used to invoke the parent class constructor. Let's see a simple example:



```
class Animal{
Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
Dog(){
super();
System.out.println("dog is created");
}
}
class TestSuper3{
public static void main(String args[]){
Dog d=new Dog();
}}
```

28. What is polymorphism? List it's type. Explain any one in detail.

Ans- Static polymorphism (Compile-time Polymorphism) Static Polymorphism in Java decides which method to execute during compile time. In static polymorphism, the object would behave differently for the same trigger. This is why multiple methods are incorporated into the same class.

Dynamic Polymorphism (run time polymorphism)- Dynamic polymorphism in Java refers to the process when a call to an overridden process is resolved at the run time. The reference variable of a superclass calls the overridden method.

29. Explain concept of method Overloading.

Ans- Method Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters, or a mixture of both.

Two or more methods can have the same name inside the same class if they accept different arguments. This feature is known as method overloading. Method overloading is achieved by either: changing the number of arguments. or changing the data type of arguments.

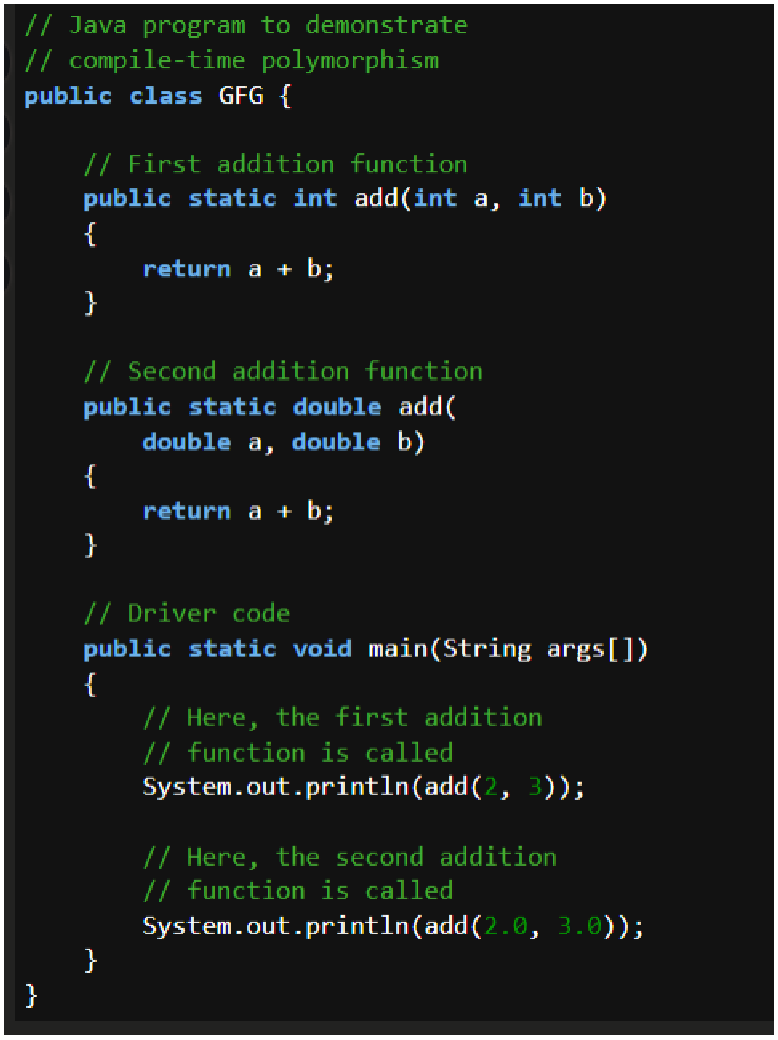
30. Explain concept of method Overriding.

Ans- Method overriding occurs when a subclass provides a particular implementation of a method declared by one of its parent classes.

Method overriding helps in writing a generic code based on the parent class. It provides multiple implementations of the same method and can be used to invoke parent class overridden methods using super keyword. It defines what behavior a class can have.

31. With the help of example explain compile time polymorphism.

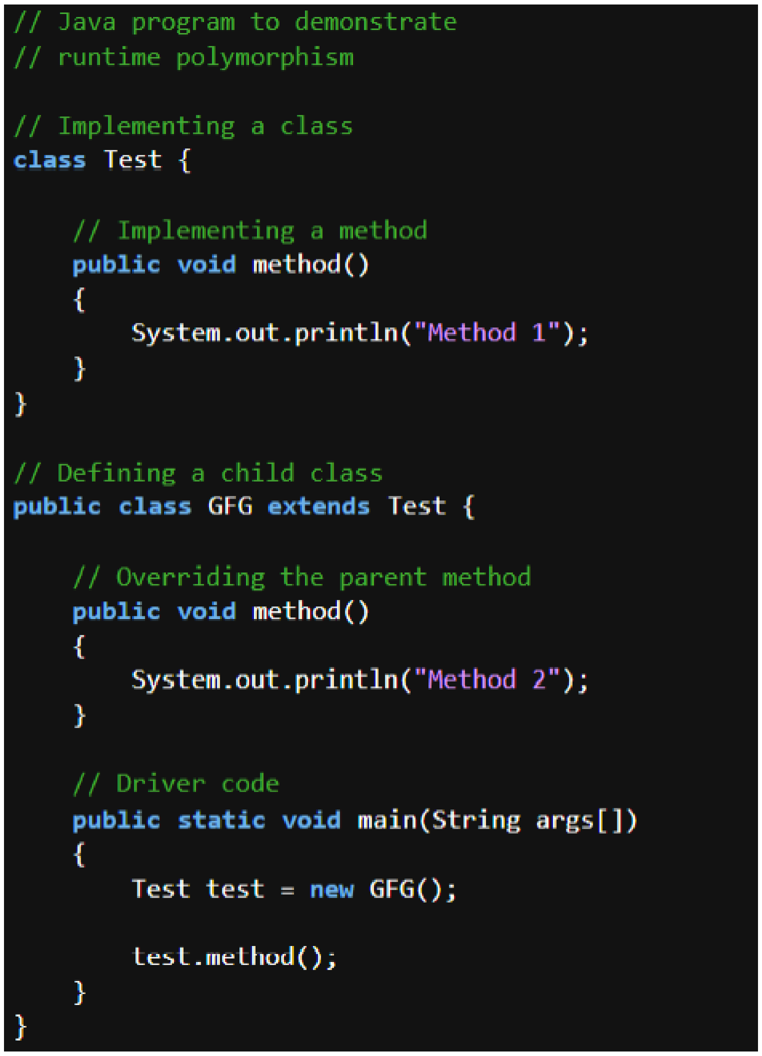
Ans-Whenever an object is bound with its functionality at the compile time, this is known as the compile-time polymorphism. At compile-time, java knows which method to call by checking the method signatures.



```java
// Java program to demonstrate
// compile-time polymorphism
public class GFG {

    // First addition function
    public static int add(int a, int b)
    {
        return a + b;
    }

    // Second addition function
    public static double add(
        double a, double b)
    {
        return a + b;
    }

    // Driver code
    public static void main(String args[])
    {
        // Here, the first addition
        // function is called
        System.out.println(add(2, 3));

        // Here, the second addition
        // function is called
        System.out.println(add(2.0, 3.0));
    }
}
```

32. With the help of example explain run time polymorphism.

Ans-Whenever an object is bound with the functionality at run time, this is known as runtime polymorphism. The runtime polymorphism can be achieved by method overriding.

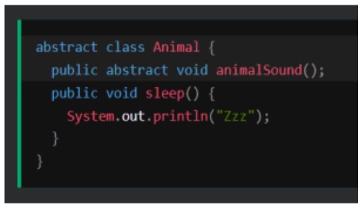


```java
// Java program to demonstrate
// runtime polymorphism

// Implementing a class
class Test {

    // Implementing a method
    public void method()
    {
        System.out.println("Method 1");
    }
}

// Defining a child class
public class GFG extends Test {

    // Overriding the parent method
    public void method()
    {
        System.out.println("Method 2");
    }

    // Driver code
    public static void main(String args[])
    {
        Test test = new GFG();

        test.method();
    }
}
```

33. What is Abstraction? What are its type.

Ans-Data abstraction is the process of hiding certain details and showing only essential information to the user.

Abstraction can be achieved with either abstract classes or interfaces



```
abstract class Animal {
  public abstract void animalSound();
  public void sleep() {
    System.out.println("Zzz");
  }
}
```

34. Explain purpose of packages.

Ans-A package in Java is used to group related classes. Think of it as a folder in a file directory.

We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories:

Built-in Packages (packages from the Java API)

User-defined Packages (create your own packages)

Built-in Packages

The Java API is a library of prewritten classes, that are free to use, included in the Java Development Environment.

The library contains components for managing input, database programming, and much much more.

35. What is package.?

Ans--Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces.

Package names and directory structure are closely related. For example if a package name is college.staff.cse, then there are three directories, college, staff and cse such that cse is present in staff and staff is present college.

36. Explain use of following predefined packages:

a. java. Lang-Provides classes that are fundamental to the design of the Java programming language. The most important classes are Object, which is the root of the class hierarchy, and Class, instances of which represent classes at run time.

b. java. Util-t contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).

c. java. Io-Java brings various Streams with its I/O package that helps the user to perform all the input-output operations. These streams support all the types of objects, data-types, characters, files etc to fully execute the I/O operations.

d. java. Sql-Provides the API for accessing and processing data stored in a data source (usually a relational database) using the Java™ programming language.

e. javax.swing-Provides interfaces that enable the development of input methods that can be used with any Java runtime environment. Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

37. List and explain different access specifiers.

Ans-There are four types of Java access modifiers:

1. Default: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
2. Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.
3. Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
4. Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

38. How packages are created and used explain.

Ans- A java package is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

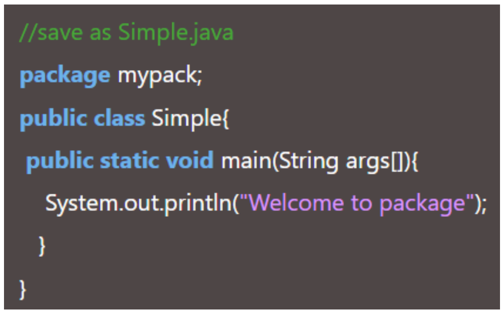
There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.

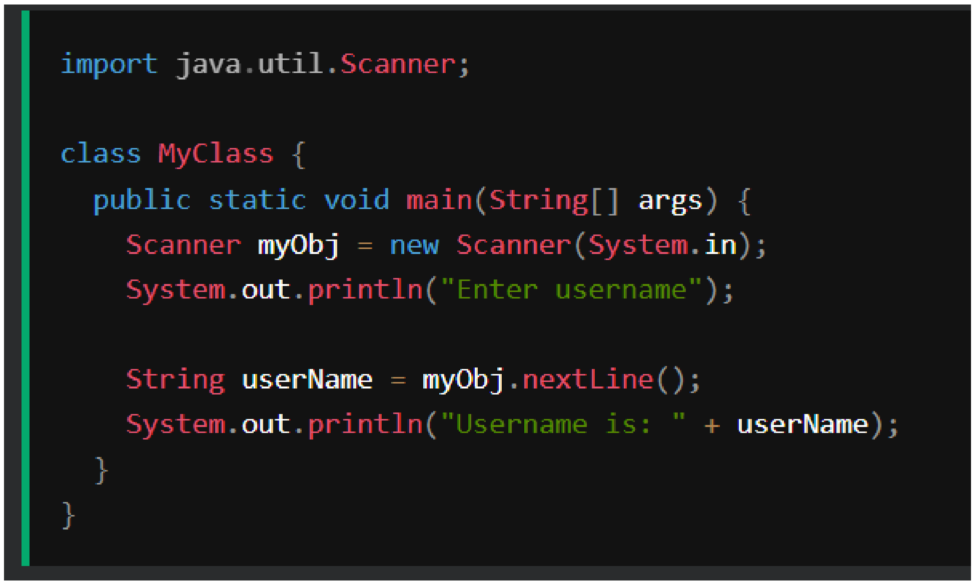


```
//save as Simple.java
package mypack;
public class Simple{
 public static void main(String args[]){
    System.out.println("Welcome to package");
   }
}
```

39. Explain purpose of import keyword.

Ans- The import keyword is used to import a package, class or interface.

Import statement in Java is helpful to take a class or all classes visible for a program specified under a package, with the help of a single statement.



```
import java.util.Scanner;

class MyClass {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);
    System.out.println("Enter username");

    String userName = myObj.nextLine();
    System.out.println("Username is: " + userName);
  }
}
```

40.What is exception? List and explain different types of exception available in

java.

Ans- An exception is an event that occurs during the execution of a program that disrupts the normal flow of the program's instructions.

There are two kinds of exceptions in Java:

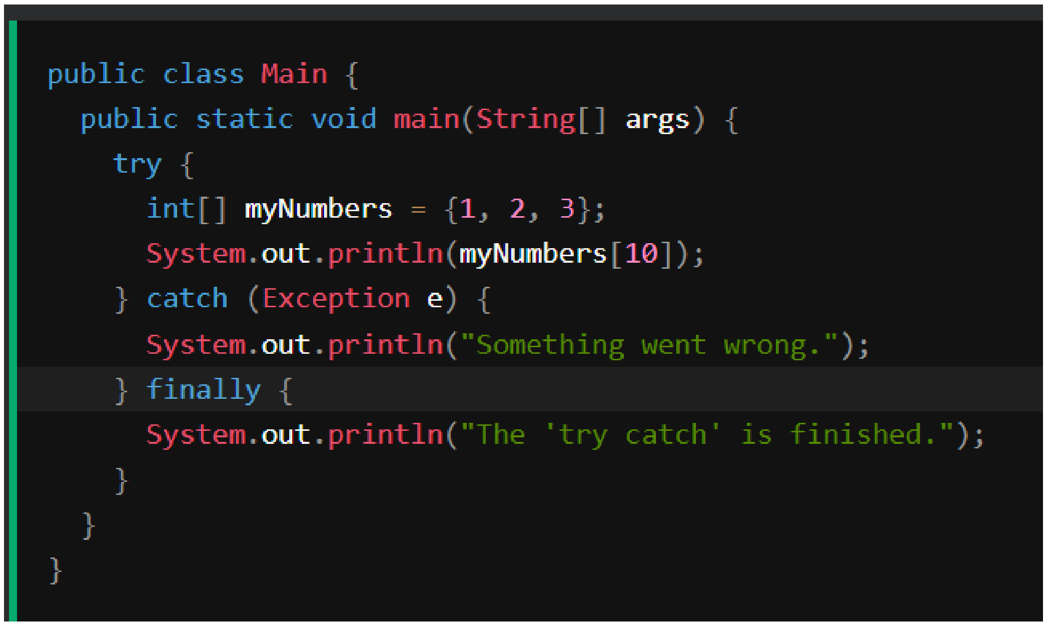
1. Checked exceptions: These are the exceptions that are checked by the compiler at compile time.
2. Unchecked exceptions: These are the exceptions that are not checked by the compiler at compile time. They include runtime exceptions and errors.

41.Explain try-catch-finally block with example.

Ans- The try statement allows you to define a block of code to be tested for errors while it is being executed.

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

The finally statement lets you execute code, after try...catch

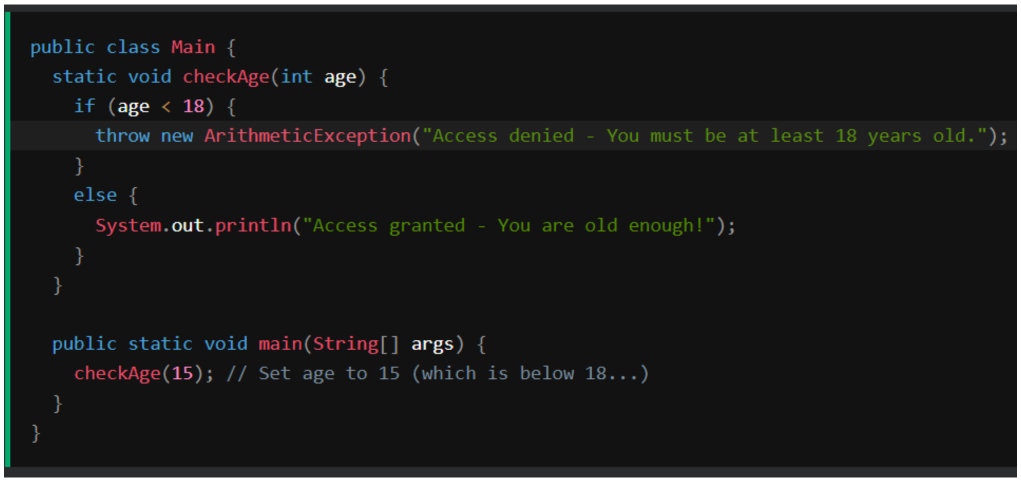


```java
public class Main {
  public static void main(String[] args) {
    try {
      int[] myNumbers = {1, 2, 3};
      System.out.println(myNumbers[10]);
    } catch (Exception e) {
      System.out.println("Something went wrong.");
    } finally {
      System.out.println("The 'try catch' is finished.");
    }
  }
}
```

42.Explain throws keyword with suitable example.

Ans- The throw statement allows you to create a custom error.

The throw statement is used together with an exception type. There are many exception types available in Java: ArithmeticException, FileNotFoundException, ArrayIndexOutOfBoundsException, SecurityException.



```java
public class Main {
  static void checkAge(int age) {
    if (age < 18) {
      throw new ArithmeticException("Access denied - You must be at least 18 years old.");
    }
    else {
      System.out.println("Access granted - You are old enough!");
    }
  }

  public static void main(String[] args) {
    checkAge(15); // Set age to 15 (which is below 18...)
  }
}
```

43. Explain throw keyword with suitable example.

Ans-Same as Ans 42 !!!

44.What is collection? What are advantages of using collection.

Ans- The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Advantages of the Collection Framework:

1. Consistent API: The API has a basic set of interfaces like Collection, Set, List, or Map, all the classes (ArrayList, LinkedList, Vector, etc) that implement these interfaces have some common set of methods.
2. Reduces programming effort: A programmer doesn't have to worry about the design of the Collection but rather he can focus on its best use in his program.
3. Increases program speed and quality: Increases performance by providing high-performance implementations of useful data structures and algorithms.

45.Define collection framework.

Ans- The Java collections framework provides a set of interfaces and classes to implement various data structures and algorithms.

For example, the LinkedList class of the collections framework provides the implementation of the doubly-linked list data structure.

46.What is List? Explain different methods of List interface.

Ans- The List interface provides a special iterator, called a ListIterator, that allows element insertion and replacement, and bidirectional access in addition to the normal operations that the Iterator interface provides. A method is provided to obtain a list iterator that starts at a specified position in the list.

47.What is ArrayList? Explain different methods of ArrayList class.

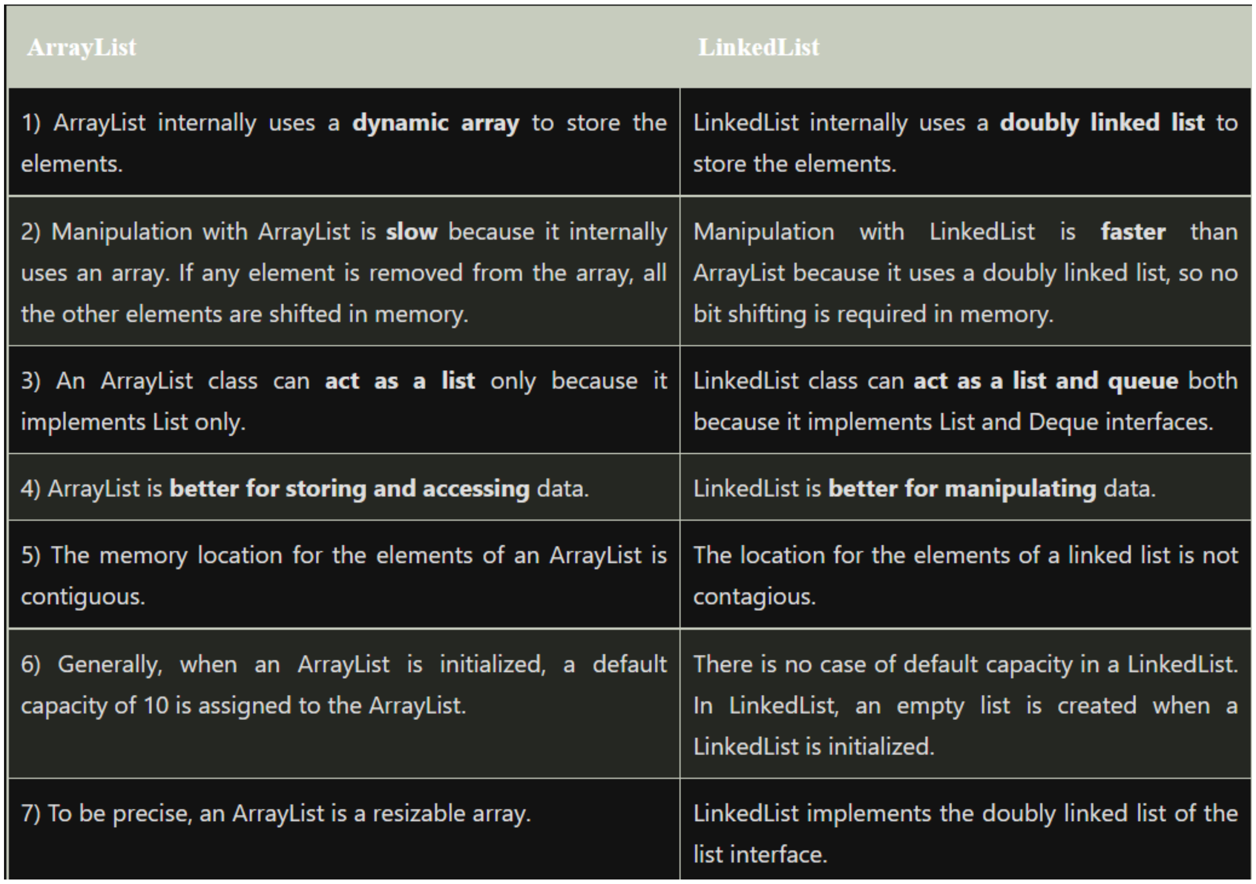
Ans- An ArrayList class is a resizable array, which is present in the java. util package. While built-in arrays have a fixed size, ArrayLists can change their size dynamically. Elements can be added and removed from an ArrayList whenever there is a need, helping the user with memory management.

48.What is LinkedList? Explain different methods of LinkedList class.

Ans- Java LinkedList class uses a doubly linked list to store the elements. It provides a linked-list data structure. It inherits the AbstractList class and implements List and Deque interfaces.

49. Write difference between ArrayList and LinkedList.

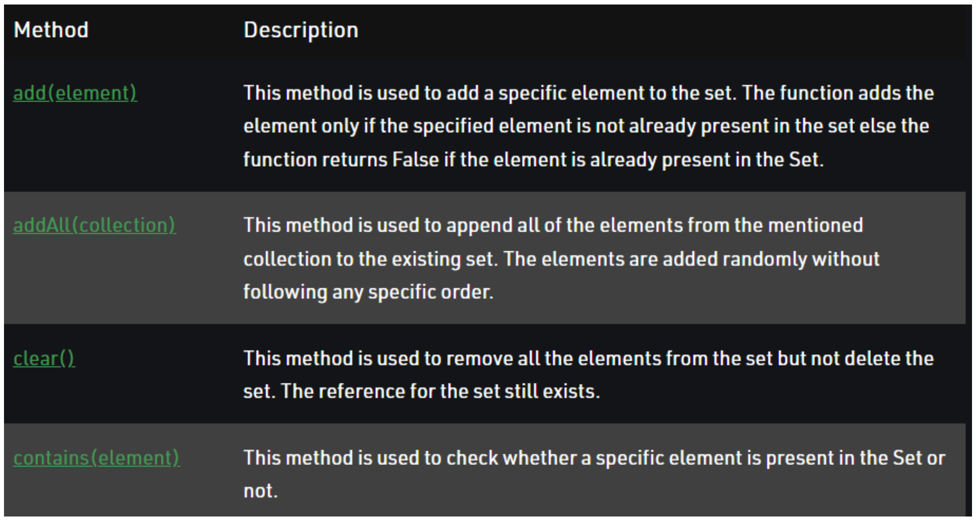
Ans-



ArrayList	LinkedList
1) ArrayList internally uses a **dynamic array** to store the elements.	LinkedList internally uses a **doubly linked list** to store the elements.
2) Manipulation with ArrayList is **slow** because it internally uses an array. If any element is removed from the array, all the other elements are shifted in memory.	Manipulation with LinkedList is **faster** than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
3) An ArrayList class can **act as a list** only because it implements List only.	LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces.
4) ArrayList is **better for storing and accessing** data.	LinkedList is **better for manipulating** data.
5) The memory location for the elements of an ArrayList is contiguous.	The location for the elements of a linked list is not contagious.
6) Generally, when an ArrayList is initialized, a default capacity of 10 is assigned to the ArrayList.	There is no case of default capacity in a LinkedList. In LinkedList, an empty list is created when a LinkedList is initialized.
7) To be precise, an ArrayList is a resizable array.	LinkedList implements the doubly linked list of the list interface.

50. What is Set? Explain different methods of Set interface.

Ans- The set interface is present in java.util package and extends the Collection interface. It is an unordered collection of objects in which duplicate values cannot be stored. It is an interface that implements the mathematical set.



Method	Description
add(element)	This method is used to add a specific element to the set. The function adds the element only if the specified element is not already present in the set else the function returns False if the element is already present in the Set.
addAll(collection)	This method is used to append all of the elements from the mentioned collection to the existing set. The elements are added randomly without following any specific order.
clear()	This method is used to remove all the elements from the set but not delete the set. The reference for the set still exists.
contains(element)	This method is used to check whether a specific element is present in the Set or not.