JAVA

- It was developed by Sun Microsystems Inc in the year 1991, later acquired by Oracle Corporation.
- It was developed by James Gosling and Patrick Naughton.
- It is a simple programming language.

JAVA

- Java makes writing, compiling, and debugging programming easy.
- It helps to create reusable code and modular programs.
- Java is a class-based, object-oriented programmin language.

JAVA

- A general-purpose programming language made for developers to write once run anywhere that is compiled Java code can run on all platforms that support Java.

- Java applications are compiled to byte code that can run on any Java Virtual Machine. The syntax of Jav is similar to c/c++.

JAVA

- It is used for:
  - Mobile applications (specially Android apps)
  - Desktop applications
  - Web applications
  - Web servers and application servers
  - Games
  - Database connection
  - And much, much more!

JAVA Terminology

- Java Virtual Machine(JVM):
  - Writing a program is done by java programmer like you and me.
  - The compilation is done by JAVAC compiler which is a primary Java compiler included in the Java development kit (JDK). It takes Java progra as input and generates bytecode as output.

JAVA Terminology

- Java Virtual Machine(JVM):
  - It is saved as .class file by the compiler
  - In Running phase of a program, JVM executes the bytecode generated by the compiler.

JAVA Terminology

- Java Development Kit(JDK):
  - It is a complete java development kit that includes everything including compiler, Java Runtime Environment (JRE), java debuggers, java docs etc.
  - For the program to execute in java, we need to install JDK in our computer in order to create, compile and run the java program.

JAVA Terminology

- Java Runtime Environment (JRE):
  - JDK includes JRE.
  - JRE installation on our computers allow the java program to run.
  - For running the java program, a computer nee JRE.

What is Package?

- 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.

What is Package?

- Packages are divided into two categories:
  - Built-in Packages (packages from the Java API)
  - User-defined Packages (create your own packages)

What is Class?

Syntax

```
import package.name.Class;   // Import a single class

import package.name.*;   // Import the whole package
```

What is Class?

- A class is an entity that determines how an object will behave and what the object will contain.
- In other words, it is a blueprint or a set of instruction to build a specific type of object.
- Contains variables(fields), methods & Obj

What is Class?

Syntax

```
class <class_name>{
    field;
    method;
  }
```

Data Types?

- Datatype is a special keyword used to allocate sufficient memory space for the data,

- In other words Data type is used for representing the data in main memory (RAM) of the computer.

Data Types?

- Data types in Java are classified into two types:
  - **Primitive : -** boolean, char, int, short, byte, long, float and double
  - **Non-primitive : -** String, Classes, Interfaces, and Arrays.

Variables?

- Variables are containers for storing data values.
- These are assigned with a data type.
- Variable is a name of memory location.

Data Types?

- **Syntax:**

```
type variable;          //Declaration
int a;

type variable = value;  //Inline Variable
boolean check = true;
```

Assign Operator

Method?

- It is a collection of statements that are grouped together to perform an operation.

- It takes some parameters, performs some computations, and then optionally returns a value (or object).

- Used to perform certain actions, and also known as functions.

Methods

- **Syntax:**

```
public class Main {
  public void myMethod(Params) {
    // code to be executed
  }
}
```

What is Encapsulation?

- Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit.

- In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

2.50

Why use Methods?

- To achieve encapsulation in Java –
  - Declare the variables of a class as private.
  - Provide public setter and getter methods to modify and view the variables values.

For Example

```
public class EncapTest {
   private String name;
   private String idNum;
   private int age;

   public int getAge() {
      return age;
   }

   public String getName() {
      return name;
   }

   public String getIdNum() {
      return idNum;
   }

   public void setAge( int newAge) {
      age = newAge;
   }

   public void setName(String newName) {
      name = newName;
   }

   public void setIdNum( String newId) {
      idNum = newId;
   }
}
```

What is Keywords?

- Keywords are the words in a language that are used for some internal process or represent some predefined actions.
- These words are therefore not allowed to use as a variable names or objects.
- Doing this will result into a compile time error.
- Therefore, also Known as Reserve words.

Some of Keywords

boolean break catch do double
else extends final Finally float
for if implements import long
instanceof int interface new super
package private protected public return

Identifiers

- Identifiers in Java are symbolic names used for identification.
- They can be a class name, variable name, method name, package name, constant name, and more.
- Keywords/Reserve words cannot be used as identifie
- Identifiers are case-sensitive.

Invalid Identifiers

- My Variable // contains a space
- 123wscube // Begins with a digit
- a+c // plus sign is not an alphanumeric character
- variable-7 // hyphen is not an alphanumeric character
- sum_&_diff //ampersand is not an alphanumeric character

Method Overloading?

- If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

- Method overloading increases the readability of the program.

- It is a Compile time Polymorphism (static).

For Example

```
public int add(String a, String b){
        return a+b;     // concatenation
}
public int add(int a, int b){
        return a+b;     // addition of two numbers
}
public int add(int a, int b, int c){
        return a+b;     // addition of three numbers
}
```

Different Params

Multiple Params

Inheritance?

- It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.
- One object acquires all the properties and behaviors of a parent object.
- It is an important part of OOPs (Object Oriented programming system).
- SubClass can reuse methods and fields of the par

For Example

```
public class superClass{
        int a=7;
        public void add(int a, int b){
                return a+b;
        }
}

public class subClass extends superClass{
        public static void main(String[] args){
                subClass s = new subClass();
                System.out.println("Programmer salary is:"+s.a);
                System.out.println("Bonus of Programmer is:"+s.add);
        }
}
```

Method Overriding?

- If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

- When a method in a subclass has the same name, same parameters or signature, and same return type as a method in its super-class, then the method in the subclass is said to override the method in the super-class.

- It is called Run time Polymorphism (dynamic).

Interface?

- An interface is a reference type in Java.
- It is similar to class.
- It is a collection of abstract methods.
- A class implements an interface, thereby inheriting the abstract methods of the interface.
- Only method signature, no body
- Interfaces specify what a class must do and not how. It is the blueprint of the class.

For Example

```
public class MammalInt implements Animal {

  public void eat() {
    System.out.println("Mammal eats");
  }

  public void travel() {
    System.out.println("Mammal travels");
  }

public static void main(String args[]) {
    MammalInt m = new MammalInt();
    m.eat();
    m.travel();
  }
}
```

```
interface Animal {
    public void eat();
    public void travel();
}
```

2.50

this?

- this is a reference variable that refers to the current object. It is a keyword in java language represents current class object.

- this keyword can be used to refer to any member of the current object from within an instance method or a constructor.

Usage of this Keyword

- this can be used to refer current class instance variable.
- this can be used to invoke current class method (implicitly)
- this() can be used to invoke current class constructor.
- this can be passed as an argument in the method call.

Usage of this Keyword

- this can be passed as argument in the constructor call.
- this can be used to return the current class instance from the method.

static

- The keyword static indicates that the particular member belongs to a type itself, rather than to an instance of that type.

- This means that only one instance of that static member is created which is shared across all instances of the class.

For Example

```
class Counter2{
        static int count=0;//will get memory only once and retain its value

Counter2(){
        count++;//incrementing the value of static variable
        System.out.println(count);
}
public static void main(String args[]){
        //creating objects
        Counter2 c1=new Counter2();
        Counter2 c2=new Counter2();
        Counter2 c3=new Counter2();
    }
}
```

final

- In Java, the final keyword can be used while declaring an entity.

- Using the final keyword means that the value can't be modified in the future.

2.00

For Example

```
class FinalVariable {

        final int var = 50;

        var = 60  //This line would give an error

}
```

2.00

For Example

```
class FinalMethod{
  final void run(){System.out.println("running");}
}
    // Final Method can be inherited but can't be overridden
class Honda extends FinalMethod{
   void run(){System.out.println("running safely with 100kmph");} // Compile
time error
   public static void main(String args[]){
   Honda honda= new Honda();
   honda.run();
   }
}
```

Conditional?

- Java uses boolean variables to evaluate conditions.

- The boolean values true and false are returned when an expression is compared or evaluated.

Conditions?

- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b
- Equal to a == b
- Not Equal to: a != b

Types of Conditional statements?

- Use **if** to specify a block of code to be executed, if a specified condition is true
- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed

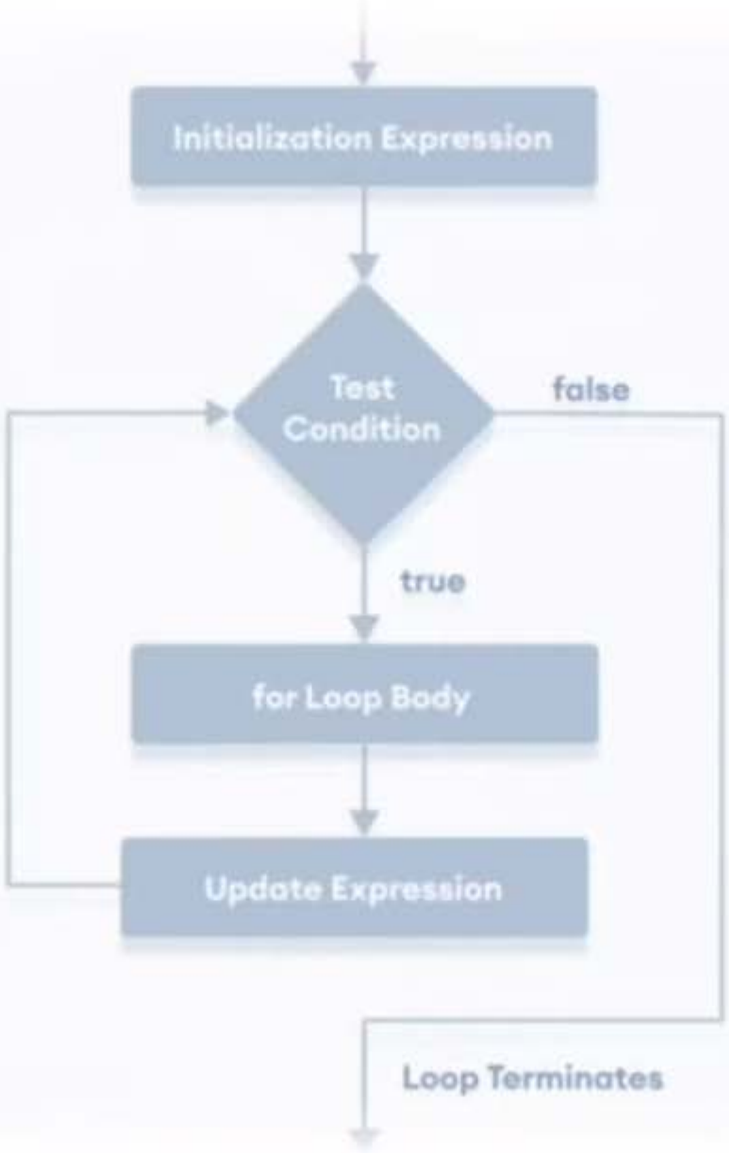
Loops?

- Loops are used to repeat a block of code.

- For example, if you want to show a message 100 times, then rather than typing the same code 100 times, you can use a loop.

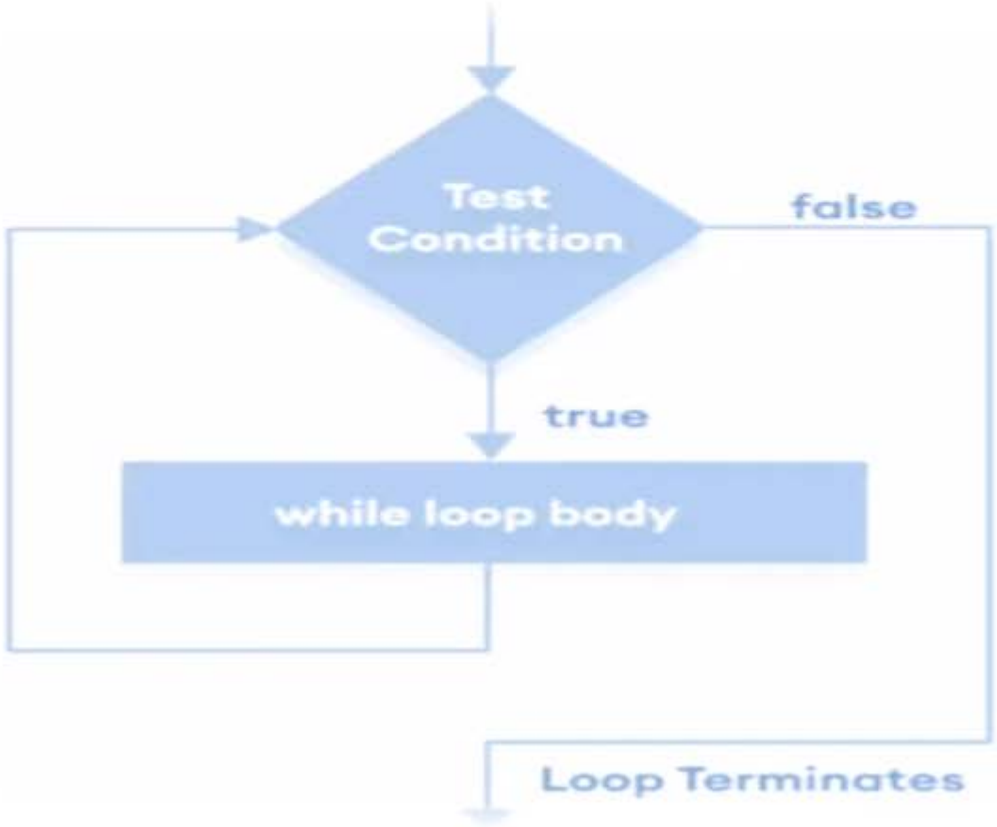
2.00

for loop



while loop

- Java while loop is used to run a specific code until a certain condition is met.



do..while loop

- The do...while loop is similar to while loop.

- However, the body of do...while loop is executed once before the test expression is checked.