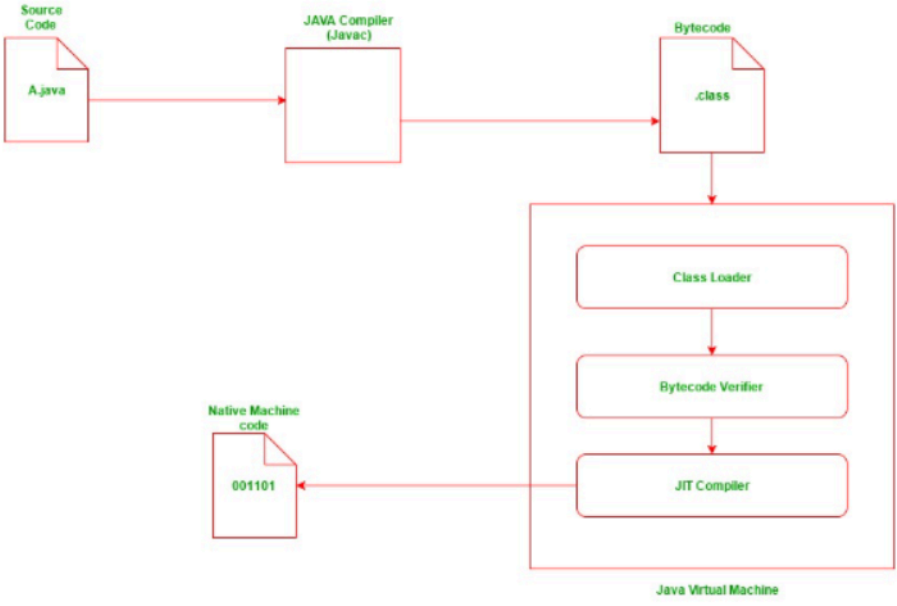
JAVA QUESTION AND NOTES

QUESTIONS

WHY IS JAVA PLATFORM INDEPENDENT ?



The JAVA compiler translates the source code into the special representation called as the BYTE CODE.
This BYTE CODE can't be run on any OS, hence it's called as architectural neutral.
In order to run this BYTE CODE, we use another software called as JAVA VIRTUAL MACHINE (or JVM). JVM is specific to every OS. JVM converts the BYTE CODE into MACHINE LANGUAGE and executes it.
Since the JAVA isn't tied to any particular machine it is called as "Platform independent".

WHAT ARE THE FEATURES OF JAVA ?

1. Platform Independence: As explained above.
2. Object-Oriented: Java implements object-oriented principles like Encapsulation, Abstraction, Inheritance, and Polymorphism.
3. Multithreaded: Multithreading in Java enables the concurrent execution of independent threads within a program, facilitating improved responsiveness and efficient utilization of system resources.
4. Portable: Java is known as a portable language because Java code can

execute on all major platforms.
5. Security: Java confines the running of code to the JVM only and does not allow it to access other parts of the computer.

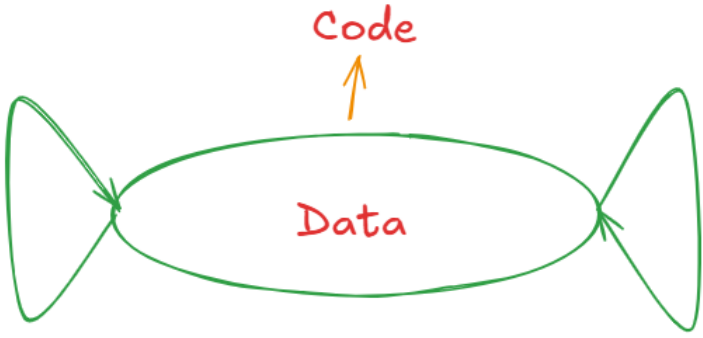
LIST THE PRINCIPLE OF OOPS

CLASS AND OBJECTS

Class is a named group of properties and functions. It is blue print from objects can be created.

Objects are real world entities that are created from the class.

ENCAPSULATION



It is mechanism that keep binds the code and data together, thus keeping it safe from outside interference.

INHERITANCE

It is mechanism from which one class inherits the properties of another class. There are only 3 types of inheritance allowed in JAVA.

POLYMORPHISM

Poly means “many” and Morphism means “taking forms”. It means objects can take different forms or multiple behavior based on the context in which they are used.

TYPE CONVERSIONS IN JAVA

WIDENING OR AUTOMATIC TYPE CONVERSIONS

The conditions to be met are:

1. The two types must be compatible
2. Destination must be larger than the source

```
int i = 3;
float f = i;
System.out.println(f)

// Output : 3.0
```

NARROWING TYPE CONVERSION

it is for incompatible data types. It happens from larger to smaller.

```
float f = 4.35;
int i = (int) f;
System.out.println(i);

// Output : 4
```

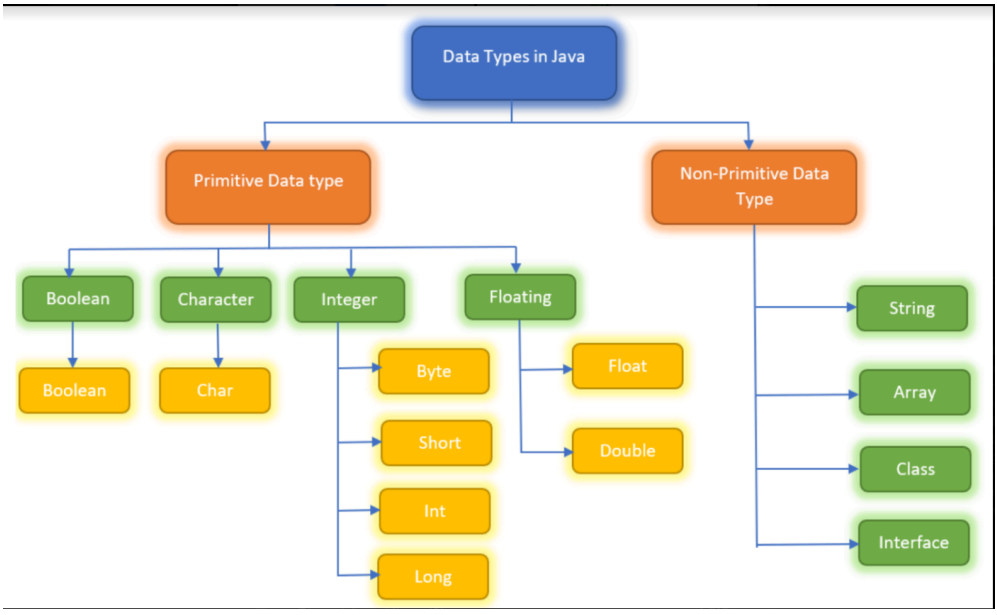
WHAT IS MEANT BY DYNAMIC DISPATCH OR RUNTIME POLYMORPHISM

Dynamic dispatch is a mechanism by which the call to a overridden method is resolved at runtime rather then compile time.

When an overridden method is called with respect to a super class reference, JAVA has to decide to which version of that method is to be executed based on the type of object being referred at the time of the call.

This is done because resolving this at runtime is faster than compile time.

LIST THE DIFFERENT TYPES OF DATA TYPES PRESENT IN JAVA



EXPLAIN THE ACCESS MODIFIERS IN JAVA

PRIVATE:

Example : The password to your house WIFI is only know the members of the house and not anyone else. (Same class only)

DEFAULT:

Example : The apartment WIFI password is know only to the members of the apartment and not anyone else outside it. (non sub-classes and sub-classes within the same package)

PROTECTED:

Example : Suppose you are hosting a house party, then the people in your apartment, your children (sub-classes) and guests (children in different apartment/package) can attend it.

PUBLIC:

Example : Anyone can ask for a glass of water from your house.

	PRIVATE	DEFAULT	PROTECTED	PUBLIC
Same class	Yes	Yes	Yes	Yes
Same package Subclass	No	Yes	Yes	Yes
Same package Non-subclass	No	Yes	Yes	Yes
Different package Subclass	No	No	Yes	Yes
Different package Non-subclass	No	No	No	Yes

TOPICS TO LEARNT FURTHER

MEMORY MANAGEMENT : HEAP AND STACK

MULTITHREADING

EXCEPTION HANDLING

COLLECTION FRAMEWORK

NOTES

WHAT IS A CLASS ?

A class is a named group of properties and functions or it is a user defined data-type.

A class isn’t a real world entity and it is just a template from which objects can be created.

The basic structure of a class in JAVA is

```
class ClassName{
   // properties
```

```
    // functions
}
```

WHAT IS AN OBJECT ?

An object is a real world entity which is created from the class.

They are real and they are present everywhere.

Objects in JAVA are created using the new operator. It's of 2 steps : declaring and instantiating.

```
ClassName cn1;        // Declaring
cn1 = new ClassName(); // Instantiating
```

WHAT ARE INSTANCE VARIABLE ?

There are the variables present inside the class but outside of any constructor or a method.

Every object has a it's own copy of variables as soon as it is initialized.

```
class ClassName{
    String name;      // instace variable
    int age;          // istance variable

    ClassName(){
        int x = 5;      // contructor variable
    }

    int function(){
        int y = 5;      // method variable
        return y
    }
}
```

WHAT ARE METHODS ?

Methods are the functions that are accessed by the objects with the help of dot operator.

```
class ClassName{
    void hello(){
        System.out.println("Hello, nice to meet you");
    }
}
```

```
ClassName cn1 = new ClassName();

cn1.hello();
```

WHAT ARE CONSTRUCTORS ?

Constructors are special functions present in the class that have the same name as that of the class and they don't have any return type.

It is initialized the objects as soon as it is created.

There are 4 types of constructors in JAVA :

1. Default : Is no constructor is specified
2. No - parameterized : has 0 params
3. Parameterized : has 1 or more params
4. Copy Constructor

```
class ClassName{
   int age;
   String name;

   ClassName(String name,int age){    // constructor
      this.name = name;
      this.age = age;
   }
}
```

WHAT IS THE THIS KEYWORD ?

The 'this' keyword refers to the current object in a method or a function.

The most common use of this keyword is :

1. It eliminates the confusion between the instance variable and the params of the constructor if they have same name.

WHAT IS OVERLOADING ?

Function or objects with the name but with different parameters or return types.

WHAT IS MEANT BY THE STATIC KEYWORD ?

When ever we create a new object, it has it's set of own instance variables. Now suppose we want all the objects to share a common variable, then we can use the word as static.

Similarly we can also make a function static so that it is accessible to all the objects of a class.

Points to be noted:

1. Static variables can only be accessed by static methods.
2. They can't refer to this or super in any way.

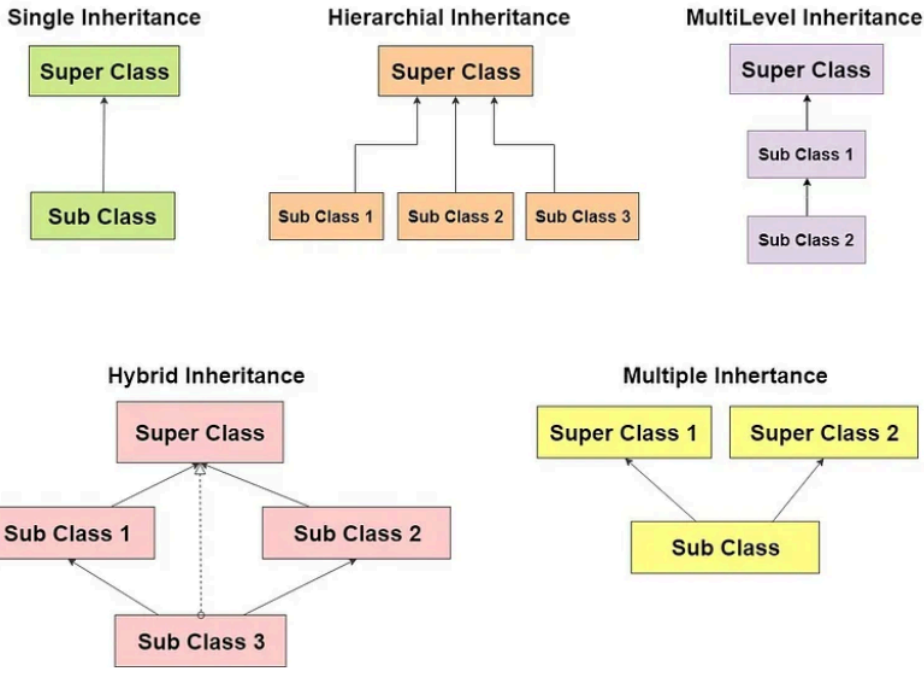
```
class Family{
    static String familyName;  // this name is same for all the objects present in the class

    String name;
    int age;

    private Family(String familyName, String name, int age){
        this.familyName = familyName;
        this.name = name;
        this.age = age;
    }

    void introduce(){
        // message
    }

}
```

WHAT IS INHERITANCE ?

Heritance is the mechanism by which one class inherits the properties of another class.

There 3 types of inheritance supported by JAVA and they are:

1. Single level
2. Multi level
3. Hierarchical level



Hybrid and Multiple inheritance aren't supported in the JAVA.

WHAT IS SUPER METHOD ?

When ever we have any type of constructor for the parent class, we must have for the child class too. In order to create a constructor for the child class we use the keyword "super". This avoid the burden of re-initialization

```
class Parent{
   float wealth;
   Parent(float wealth){
      this.wealth = wealth;
   }
}

class Child extends Parent{
   Child(float wealth){
      super(wealth);
   }
}
```

WHAT IS UPCASTING ?

```
class P{
```

```
}

class C extends P{

}

P p = new C();     // this is called upcasting
```

Here we are creating a reference of the Parent class that is pointing to the object of the child class, this is called as upcasting.

Every instance of a child is an instance of it’s parent.

The reference can access the methods and the variables present in the parent class too.

WHAT IS OVERRIDING ?

```
class Animal{
   public void growl(){
      System.out.println("General Animal Sound");
   }

   class
}

class Lion extends Animal{
   public void growl(){
      System.out.println("Grrrrrr");
   }
}

Lion lion = new Lion();

lion.growl();

// Output
// Grrrrrr
```

When a parent class and a child class have a method of same name and return type, then when the method is called from the child class it simply overrides the parent method.

When class has final keyword before it’s method, then it can’t be overridden by the child class

WHAT DOES THE FINAL KEYWORD DO ?

The final keyword is used to indicate that a variable can’t be changed once initialized, a method can’t be overridden in the child class and a

class can’t be inherited by any other class.

WHAT IS ABSTRACT METHOD AND CLASS ?

We know that the word final will make prevent the method from being overridden but the word “abstract” will make the particular method to be overridden compulsorily.

Any class containing the abstract method shall have the abstract keyword before the class name too.

Any class that is abstract in nature, can’t have it’s objects initialized.

Abstract classes can have constructors.

```
abstract class Animal {
   abstract public void growl();

   public static void main(String[] args) {
      Lion lion = new Lion();
      lion.growl();
   }
}

class Lion extends Animal{
   public void growl(){
      System.out.println("Grrrrrrr");
   }
}
```

What are interfaces in JAVA ?

In order to use multiple inheritance freely in Java, we use interfaces.

🔖 **Note:**

1. Interfaces **cannot have objects** of their own.
2. They are **implemented using the** `implements` **keyword**.
3. By default, **all methods are abstract** (except those declared with `default`, `static`, or `private` – from Java 8+).
4. Interfaces allow a class to **inherit behavior from multiple sources**, avoiding the diamond problem.