**Core Java**

* One of the most famous object oriented programming language used in these times
* We can simple start from term API (Application Programming Interface) this is the collection of packages. Where a package is a collection of classes, interfaces and a sub-package where a sub-package is collection of classes, interfaces and sub sub-package.
* **Class:** Class is a blue print which is containing only list of variables and method and no memory is allocated for them. A class is a group of objects that has common properties. Generally, it contains data members, Methods, constructors, block, class and interface.
* **Object:** object is an instance of a class. Object have state, behavior and identity. State represents the data of the object, Behavior represents the functionality of the object, Identity is unique id for that object identification not visible to user but used internally by compiler.
* **Block:** Set of statements that are written in between the curly brackets which are executed from the top to bottom without any conditions.

**{**

**statement 1**

**statement 2**

**statement 3**

**-**

**-**

**statement n**

**}**

* **Data Types in Java**: Data Type can be said as the keyword that represent the what type of data is being stored in particular variable. There are total 8 primitive data types in java.
  + Boolean: A variable which stores only one bit of data that is wither 0 or 1 which generally represents false and true, respectively.
  + Byte: A variable which can store total for 8 bits of data. In decimal, it can store from -128 to +128
  + Int: variables which store integer values which can be accommodated in the 32 bits of system memory.
  + Short: variables which store integer values which can be accommodated in the 16 bits of memory.
  + Long: variables which store integer values which can be accommodated in the 64 bits of memory.
  + Float: variables which store floating point values which can be accommodated in the 32 bits.
  + Double: variables which store floating point values which can be accommodated in the 64 bits.
  + Char: Which store 16 bit of code (2 bytes) who’s each combination represents a Unicode character.
* **Variable**: A part of memory that contain a certain data of a particular data type with its identification. Non-static variable (Instance), Static variable, Class variable, Local Variable, parameters.
* All variables must have a type. You can use primitive types such as int, float, boolean, etc. Or you can use reference types, such as strings, arrays, or objects.

static int numGears = 6;

boolean result = true;

char capital = 'C';

byte b = 100;

short s = 10000;

int i = 100000;

* **Array:** It a group of data elements with similar data types stored in a consecutive memory location. Where each element can be accessed by its index. Index of an array always starts from 0.

int[] anArray;

anArray[0] = 100;

anArray[1] = 10;

anArray[2] = 12;

anArray = new float[10];

* Operators:

Arithmetic: +(Addition), – (Subtraction), \*(Multiplication), /(Division), %(Modulus).

Unary operator: Positive no (+), Negative no(-), increment (++),decrement (--),not (!).

Conditional: (==) equal to, (!=) not equal to,( >)greater than,( >=)greater than or equal to,(<)less than,(<=) less than or equal to.

Bitwise: & operator performs a bitwise AND operation.

^ operator performs a bitwise exclusive OR operation.

| operator performs a bitwise inclusive OR operation.

* Control Flow Statements:
  + **(If and else if)**

**If**(Conditional statements){

Block

}

**else if** (condition){

Block

}

..

..

**else**{

Block

}

* + **Switch**

**switch** (Variable) {

case 1 : statement ;

return;

case 2 : statement ;

return;

..

..

case n : statement ;

return;

* + **While : Statements are executed only after Evaluating the expression**

**while** (expression) {

statement(s)

}

* + **do-While : Evaluates the expression after the execution (at least one time statements are executed)**

**do** {

statement(s)

} **while** (expression);

**for loop:**

**for ( initiation; condition; increment or decrement;)**

**{**

**statements..**

**..**

**..**

**}**

* **Declaring Classes**

class *MyClass* {

// field, constructor, and

// method declarations

}

In general, class declarations can include these components, in order:

1. Modifiers such as public, private, and a number of others that you will encounter later.
2. The class name, with the initial letter capitalized by convention.
3. The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
4. A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
5. The class body, surrounded by braces, {}.

## **Access Modifiers**

The first (left-most) modifier used lets you control what other classes have access to a member field.

public modifier—the field is accessible from all classes.

private modifier—the field is accessible only within its own class.

## **Declaring Method**

public double cals (double wingSpan, double grossTons) {

//do the calculation here

}

More generally, method declarations have six components, in order:

1. Modifiers—such as public, private, and others you will learn about later.
2. The return type—the data type of the value returned by the method, or void if the method does not return a value.
3. The method name—the rules for field names apply to method names as well, but the convention is a little different.
4. The parameter list in parenthesis—a comma-delimited list of input parameters, preceded by their data types, enclosed by parentheses, (). If there are no parameters, you must use empty parentheses.
5. An exception list—to be discussed later.
6. The method body, enclosed between braces—the method's code, including the declaration of local variables, goes here.

* **Method Overloading:** Java can distinguish between methods with different method signatures. This means that methods within a class can have the same name if they have different parameter.

public class DataArtist {

...

public void draw(String s) {

...

}

public void draw(int i) {

...

}

public void draw(double f) {

...

}

public void draw(int i, double f) {

...

}

}

Overloaded methods are differentiated by the number and the type of the arguments passed into the method. In the code sample, draw(String s) and draw(int i) are distinct and unique methods because they require different argument types.

* **Constructor**: A class contains constructors that are invoked to create objects from the class blueprint. Constructor declarations look like method declarations—except that they use the name of the class and have no return type.

public Bicycle(int startCadence, int startSpeed, int startGear) {

gear = startGear;

cadence = startCadence;

speed = startSpeed;

}

Bicycle myBike = new Bicycle(30, 0, 8);

* **This keyword:**  With in an method or a constructor, this is a reference to current object, he object whose method or constructor is being called. You can refer to any member of the current object from within an instance method or a constructor by using this.
* **Abstract Class:**  A class which contain may contain abstract methods where **Abstract Methods** are those methods whose deceleration is present and body is not present. These classes cant be instantiated.
* **Inheritance:** Inheritance can be defined as the process where one class acquires the properties of another. The class which inherits the properties of other is known as subclass and the class whose properties are inherited is known as superclass. **extends** is the keyword used to inherit the properties of a class.
* Multiple inheritance can’t we done.

Class super{

……

…..}

class sub extends super {

….

……}

* **Interfaces:** 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.

Public interface example {

Public void example1();

Public void example2(int a, int b);

}

* + A class that implements the interface will use implements keyword and this class will provide the implementation (body) for the abstract methods that are present in the interface.
  + If these are not implemented then this class can be declared as an abstract class.
  + If not methods in the interface can also be static or default methods.
  + Multiple implementations are possible.
* **Method overriding:** Generally, subclass can implement the methods of the super class. Changing the functionality those classes can be said as the method overriding.
* **Final Keyword:**  Any variable or a method is declared as the final then those can’t be overridden or changed. If a class is declared as final they can’t be extended (or inherited).
* **Number classes :** Generally, when working with number we use data types variables. When working in real time we work with object so, we are provided with some classes called wrapper classes. Which helps us to create the object of the respective data type. And this classes provide us lot of methods which helps us to convert the object to data type and vice versa. And some other methods to find max and min values, comparing two objects, converting a string into integer and lot.
* **Autoboxing and Unboxing :** Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes this reverse conversion is unboxing.
  + **Example int to Integer**

**int x = 123;**

**Integer y = x;(Autoboxing)**

**Int z = y;(unboxing)**

* **Exceptions:** An exception is an event that occurs during the execution of a program that disturbs the flow of statements.

These exceptions can be handled using the **try and catch block**

The first step in constructing an exception handler is to enclose the code that might throw an exception within a try block.

Try {

….

…

}

catch( exception type name){

}

catch( exception type name){

}

…

finally {

}

Java try block is used to enclose the code that might throw an exception. It must be used within the method.

Java try block must be followed by either catch or finally block.

The finally block *always* executes when the try block exits. This ensures that the finally block is executed even if an unexpected exception occurs. But finally is useful for more than just exception handling.