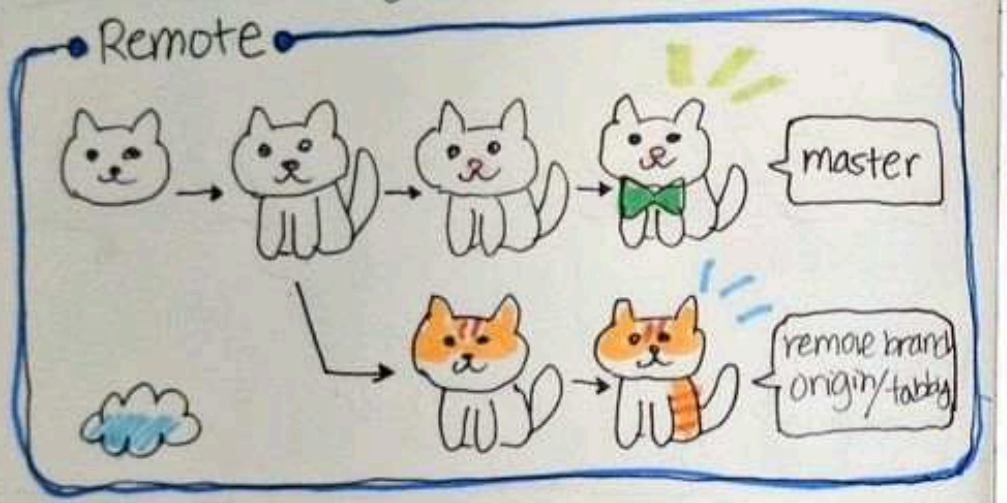
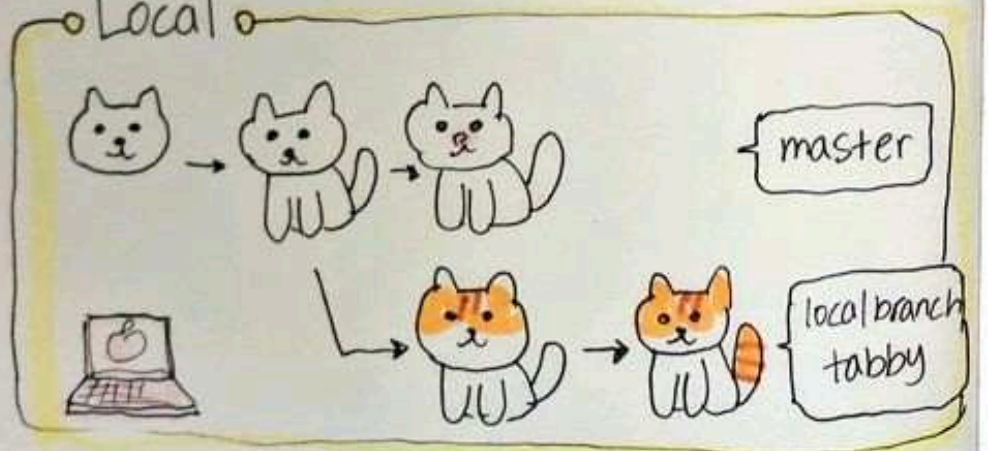
git Pull

purr

@girlie_mac

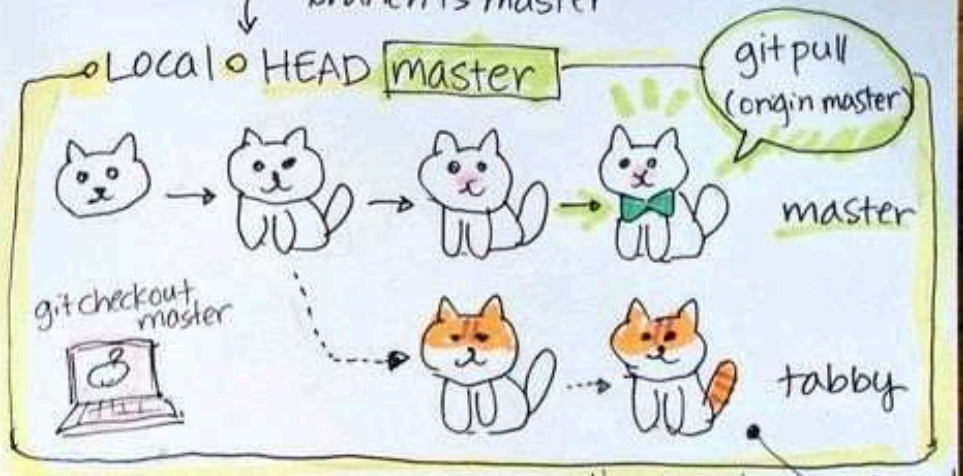
git pull updates your current HEAD branch w/ the latest Δs from remote





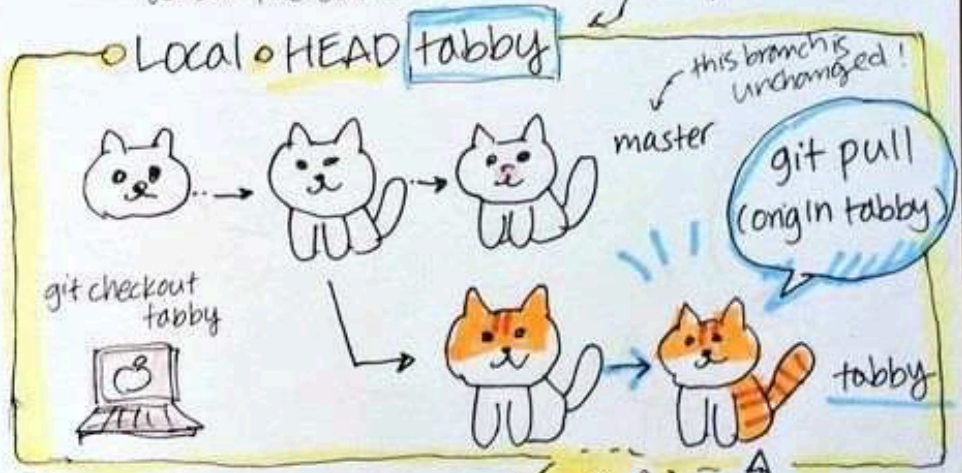
Now I'm going to 'git pull' to update a branch!

When the current branch is master



This is same as:
- git fetch origin
- git merge origin/master

When the current branch is tabby



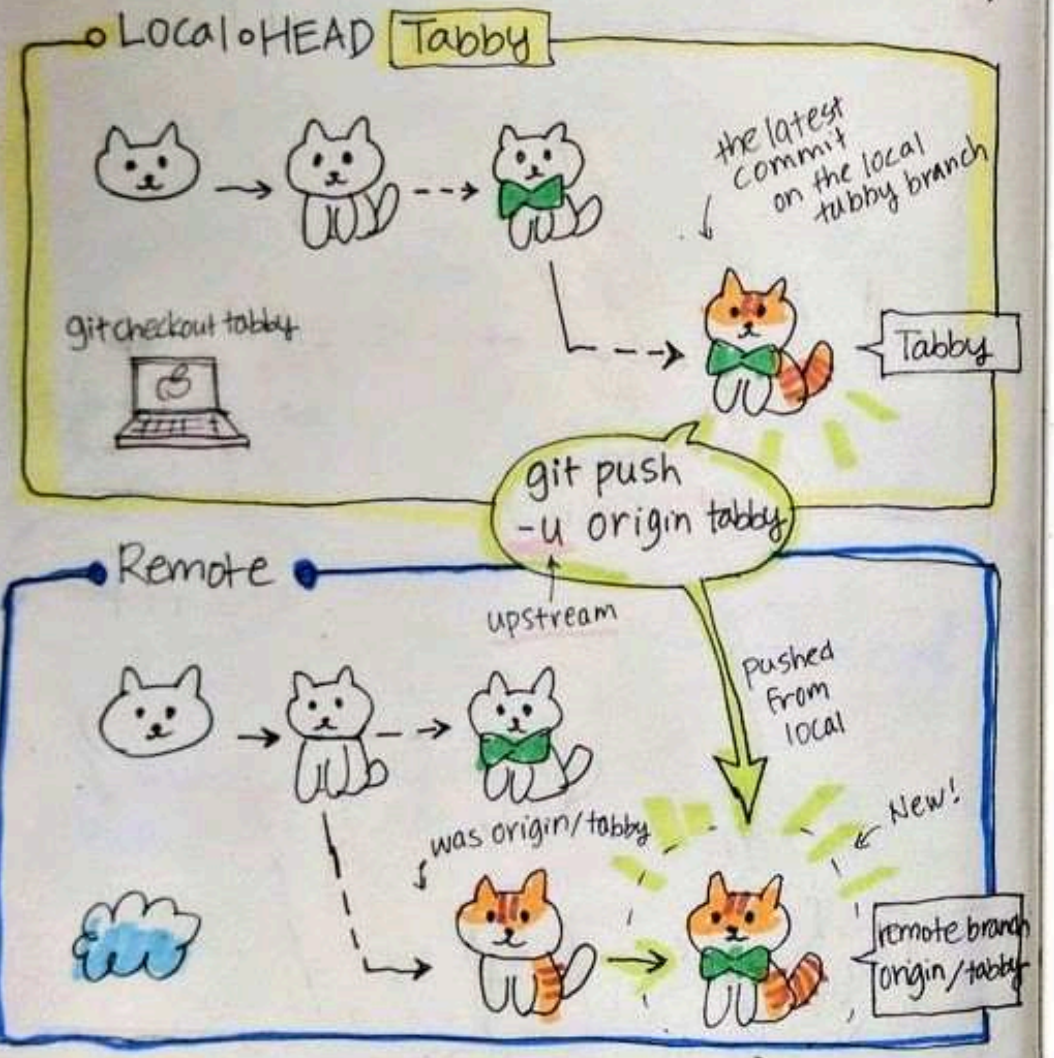
Same as:
- git fetch origin
- git merge origin/tabby

git Push

puss

@girlie_mac

♥ git push transfers your commits from your local repo to a remote repo!



Now the tabby branch on local + remote are sync'd!

♥ git push

- `git push <remote> <branch>`
  - e.g. origin
  - e.g. tabby

  push the specified branch w/ all commits

- `git push -u <remote> <branch>`
  - ↳ = --set-upstream

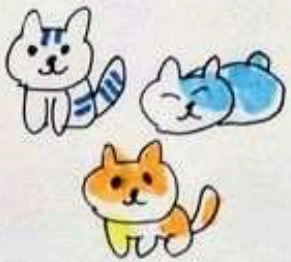
  the -u flag sets up the association between your branch + the remote branch explicitly.
  you don't need it once you've done!

- `git push -f <remote> <branch>`
  - ↳ --force

  force the push, even if it results in a non-fast-forward merge.
  just ignore + push!

- `git push --all <remote>`

  push all of your local branches!

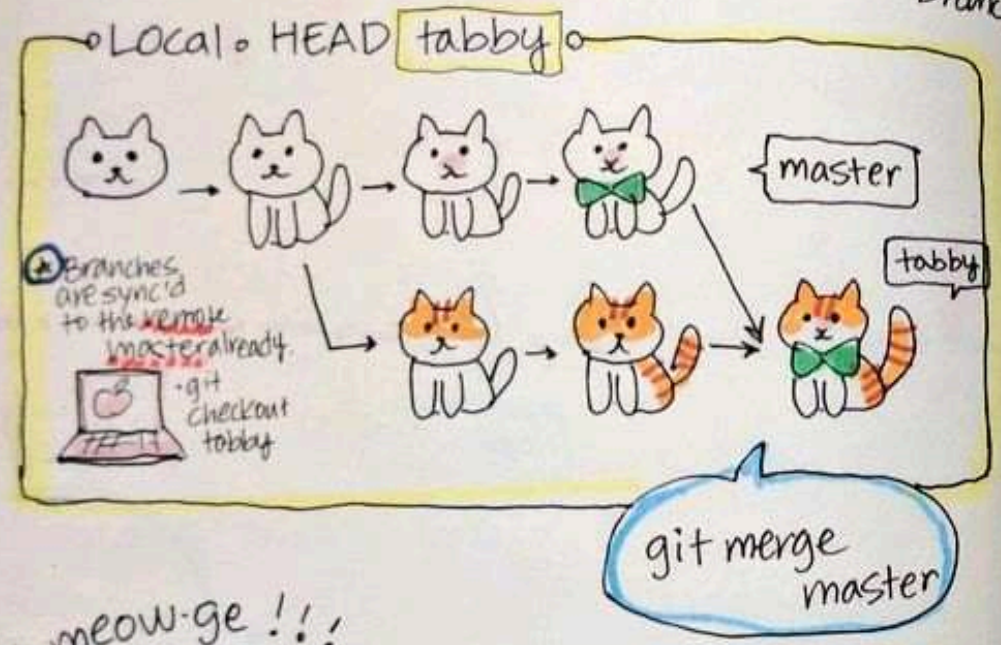


git merge

meow! ge

@girlie_mac

♥ git merge incorporates changes into the current branch.



git meow-ge !!!

★ merge is like having 2 parents & 1 resulting child !

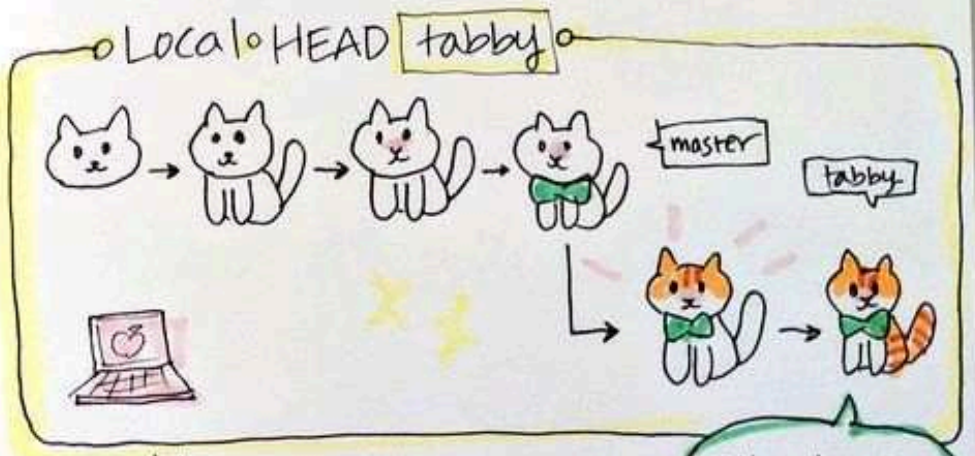
★ rebase adds all new △s on top of one parent.

PAWSOME!

git rebase

@girlie_mac

♥ git rebase moves a branch from one commit to another.



Cleaner commit history!

git rebase master

If you want to rebase from the remote master instead of local. do either:

$ git fetch origin
$ git rebase origin/master

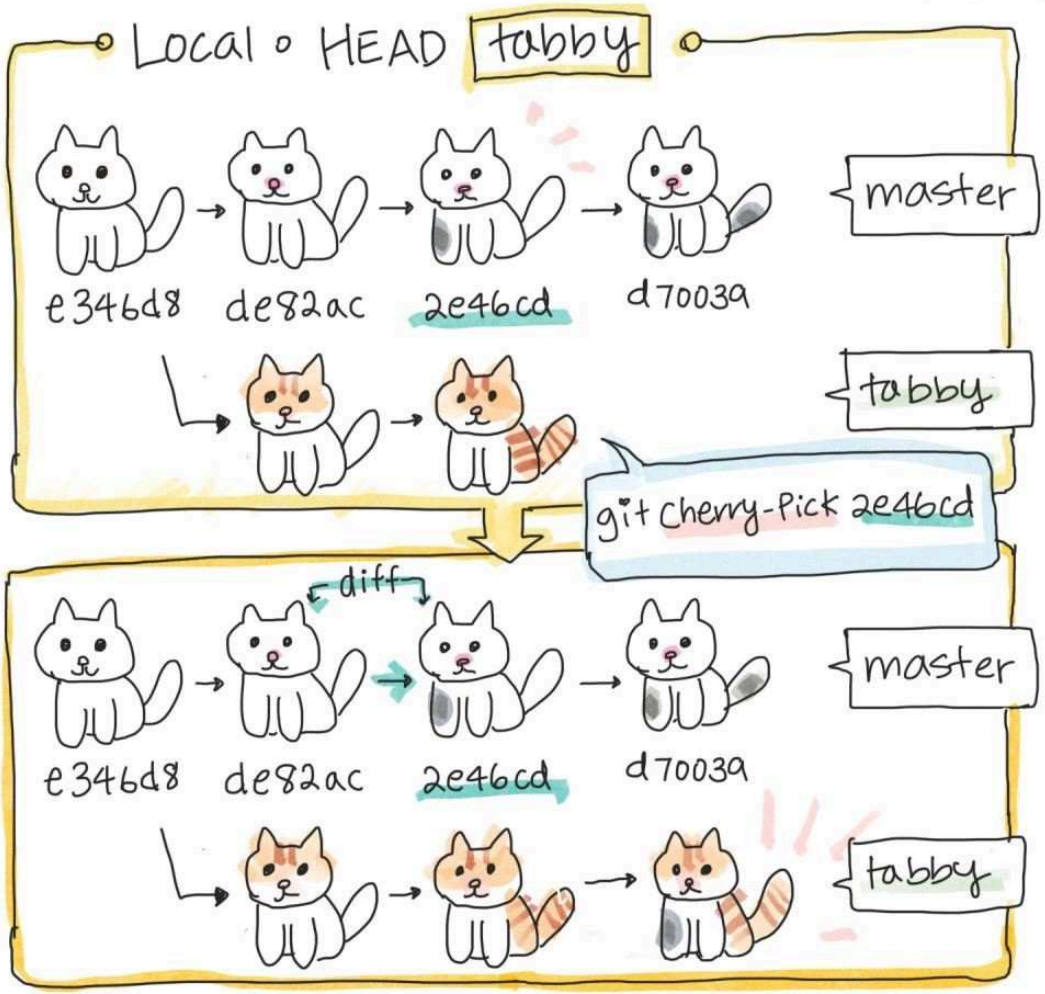
or

$ git pull --rebase origin master

git cherry-pick

@girlie_mac

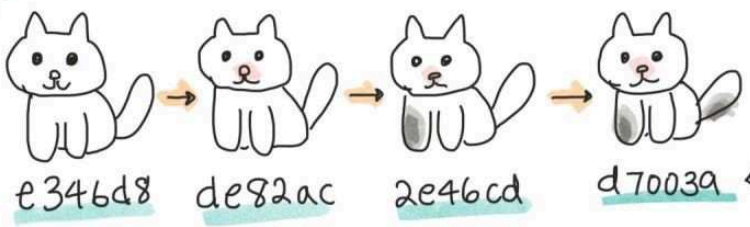
♥ git cherry-pick lets you grab some commits from one branch & apply it into another branch!



git log

♥ git log lets you view the commit history

@girlie_mac



♥ git log prints out

```
Commit e346d8
Author: Little-princess Leia <leia@kitty.cat>
Date : Sun Sept 23 17:30:42 2018 -0700
    Add body
commit de82ac
Author; Jamie <jam@kitty.cat>
Date :
....
```

git cherry-pick this SHA into the tabby branch

Simpler log with:

♥ git log --oneline

```
e346d8 Add body
de82ac Edit nose
2e46cd Add gray dot on leg
```

Git & GitHub 101

Basic CLI Commands

1. To list all files or folder in a folder

   `ls`

2. Make a new folder

   `mkdir folder_name`

3. Go inside a folder

   `cd folder_name`

4. To delete a whole non-empty directory/folder

   `rm directory_name -rf`

5. Write a file in Git Bash Vim

   `vim file_name`

   1. use insert key to enable the writing mode in any file
   2. then after finishing edits, press the left-right arrow key to disable the writing mode and then write `:x` to exit out

6. Copy + Paste in CLI

   1. Use the insert key to paste in CLI or highlight the statement then right click and copy that statemen and then right-click on CLI shows the options.

Basic Git Commands

1. To make a new file

   `touch names.txt`

2. To check if git is installed in your PC

   `git`

3. To initialize an empty Git repository in your folder

   `git init`

4. To view the changes or the untracked files in the project that's not been saved yet

   `git status`

5. Staging the files

   `git add file_name` or `git add .` (to stage everything in the current folder)

6. Committing the files

Working with Existing Projects on GitHub

Use Git Bash for Windows.

You can't directly change the contents of a repo unless you have access to it. To solve this, you create a copy (fork) of this project in your own account. In our own copy, we can do anything we want with it. After forking, we:

1. Cloning the forked project to local machine

   `git clone forked_repo_url`

2. The public repo that we forked out local copy from is known as the upstream url. We can save it as

   `git remote add upstream insert_upstream_url`

3. Creating a new branch

   `git branch branch_name`

4. Then shift the head to the above branch using the checkout command

5. Then stage. Then commit.

6. Then push. We can't push to upstream (no access). Can push to our forked repo though (origin)

   `git push origin your_branch_name`

7. Always make different branches for different pull requests if you're working on different features. 1 branch = 1 pull request (never commit on main (2))

8. To remove a commit

   1. we can remove a commit with the reset command Now it's unstaged.
   2. then add. to stage the remaining files
   3. then we can use the stash command to stash it elsewhere
   4. then, we'll have to force push this branch since the online repo contains a commit which the local repo does not

   `git push origin your_branch_name -f`

9. To make forked project even (updated) with the main project

```
git commit -m "your_message_here"
```

7. To unstage or remove a file from the staging level

   `git restore --staged file_name.txt`

8. To view the entire history of the project

   `git log`

9. Removing a commit from the history of a project

   `git reset insert_commit_hash_id_to_which_you_want_to_go_back_to_here` (all the commits or changes before this will go back to the unstaged area now)

10. After you stage a few files but then you want to have a clean codebase or reuse those files later, we can stash those changes to go back to the commit before they were staged

    `git stash`

11. Bringing back those changes or pop them from the stash

    `git stash pop`

12. To clear the changes or files in your stash

    `git stash clear`

How Git works

1. Connecting your Remote Repository to Local Repository

   `git remote add origin insert_https_project_link_here`

2. Pushing local changes to remote repository

   `git push origin master` (we're pushing to the url origin, and the branch master)

3. To view all your remote urls

   `git remote -v`

4. Never commit on the main branch since it's the one used by the people, to prevent any mishaps

5. Shifting the head to a branch (head is the pointer which points to where all you do your changes)

   `git checkout branch_name`

6. Merging your branch to main of project

   `git merge branch_name`

1. Shift the head to your main branch

   `git checkout main`

2. Fetching all the commits/changes from the main project (upstream)

   `git fetch --all --prune` (here prune gets deleted commits too)

3. Reset the main branch of origin (forked) to main branch of upstream (main project)

   `git reset --hard upstream/main`

4. Check and verify your changes

   `git log` *click q for exit from log*

5. Then push all these local changes to your online forked repo

   `git push origin main`

Method 2

1. To fetch all **at once**

   `git pull upstream main`

2. Then push to the origin url or your forked project

   `git push origin main`

Method 3

1. Update using the Fetch Upsteam button on forked repo

10. Squashing all your multiple commits into one commit

    `git rebase -i insert_hash_code_of_commit_above_which_all_your_required_co`

    If there's 4 commits. Keep 1 as the pick and then `s` or squash the other 3 into that one

11. Merge conflicts and how to resolve them

    1. They happen when multiple users edit the same code line and then push it. Git won't know which one to merge and then there'd be a conflict
    2. This has to be resolved manually by repo maintainer

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/94dcc4e7-259a-4c09-bf05-d1ce5eb2d9e3/atlassian-git-cheatsheet.pdf

LECTURE – 1

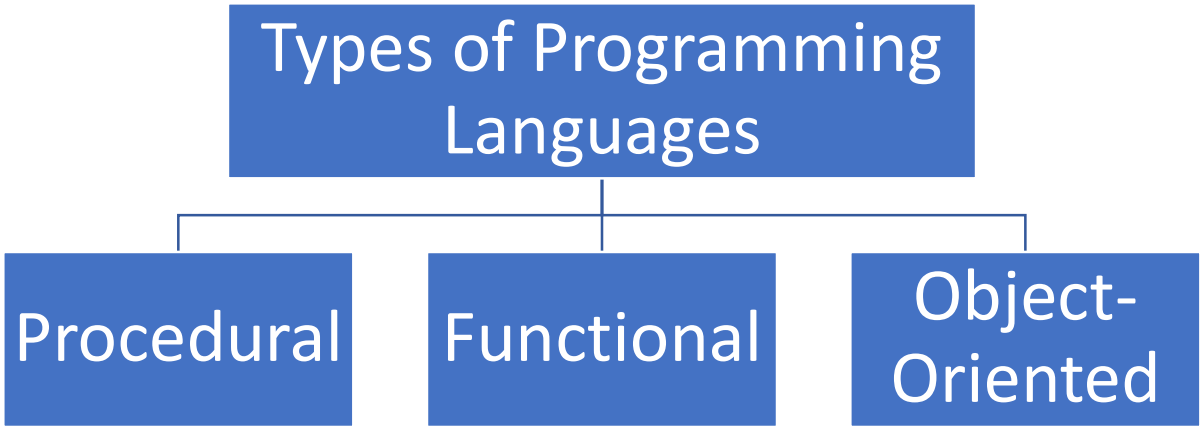
Introduction to programming

Programming is a way to instruct the computer to perform various task.

Computers only understands Binary i.e., 0's and 1's.

Instructing computers in Binary i.e. 0's and 1's are very difficult for humans so, to solve this issue we have programming languages.

Programming language: - It is a computer language used by programmers to communicate with computers.



Procedural

- Specifies a series of well-structured steps and procedures to compose a program.
- Contains a systematic order of statements functions and commands to complete a task.

Functional

- Writing a program only in pure functions i.e., never modify variables but only create new ones as an output.
- Used in a situation where we have to perform lots of different operations on the same set of data like ML.

Object Oriented

- Revolves around objects.
- Code + Data = objects
- Developed to make it easier to develop, debug, reuse and maintain software.

"One programming language can be of all 3 types like- Python"

Java Follows procedural and object oriented both types

Static VS Dynamic Languages

Static	Dynamic
Perform type checking at compile time	Perform type checking at runtime
Errors will show at compile time	Error might not show till programs run
Declare datatypes before use	No need to declare datatype of variables
More control	Saves time in writing code but might give error at runtime.

Memory Management

There are 2 types of memory Stack and Heap

When we declare a variable then the reference variable stored in stack memory points to the object of that variable stored in heap memory.

For ex:- $a = 10$

Here "a" is called reference variable, and "10" is the object of That reference variable

- Reference variable are stored in stack memory.
- Heap memory stores the objects of reference variable.

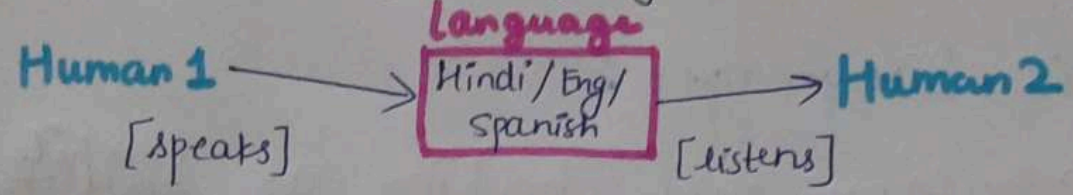
Points to remember:-

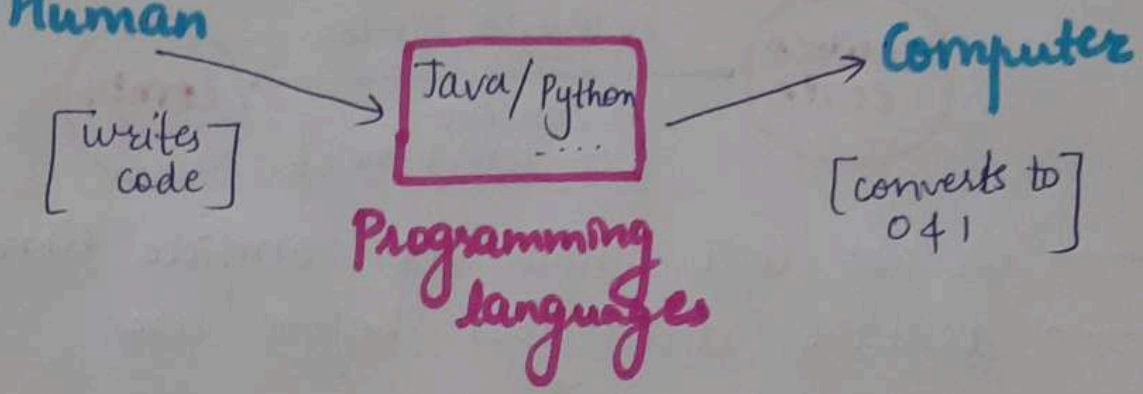
- More than one reference variable can points to the same object.
- If any changes made to the object of any reference variable that will be reflected to all others variable pointing to same object.
- If there is an object without reference variable then object will be destroyed by ***"Garbage Collection"***

2|8|21

Introduction to Programming Language

- Computers at very minute level only understands zeros & one's (0's & 1's)
- What is programming language?





- Types of Programming Languages :

Procedural :

→ series of well-structured steps & procedures to compose a program

→ contains a systematic order of statements functions and commands to complete a task.

Functional :

→ Writing a program only in pure functions i.e; never modify variables but only create new ones as an output

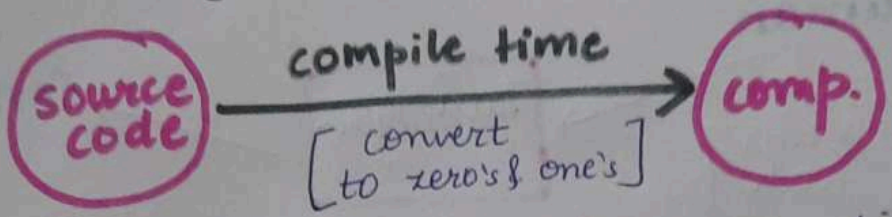
→ Used in a situation where we have to perform lots of different operations on the same set of data like ML.

Object Oriented:

→ Revolves around objects
→ code + data = objects
→ developed to make it easier to develop, debug, reuse & maintain.

• Static Languages:

→ Perform type checking at ~~runtime~~ compile time.



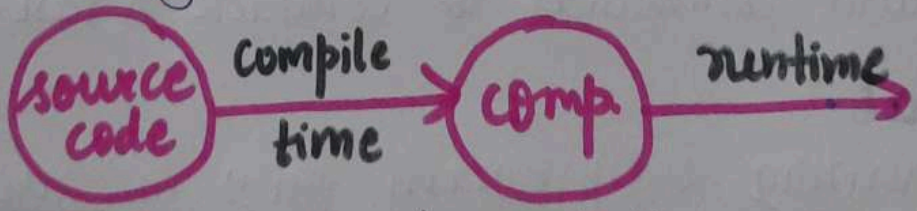
→ errors will show at compile time
→ declare datatypes before use

```
int a = 10
```

→ More control over the program.

• Dynamic Languages:

→ Performs type checking at runtime



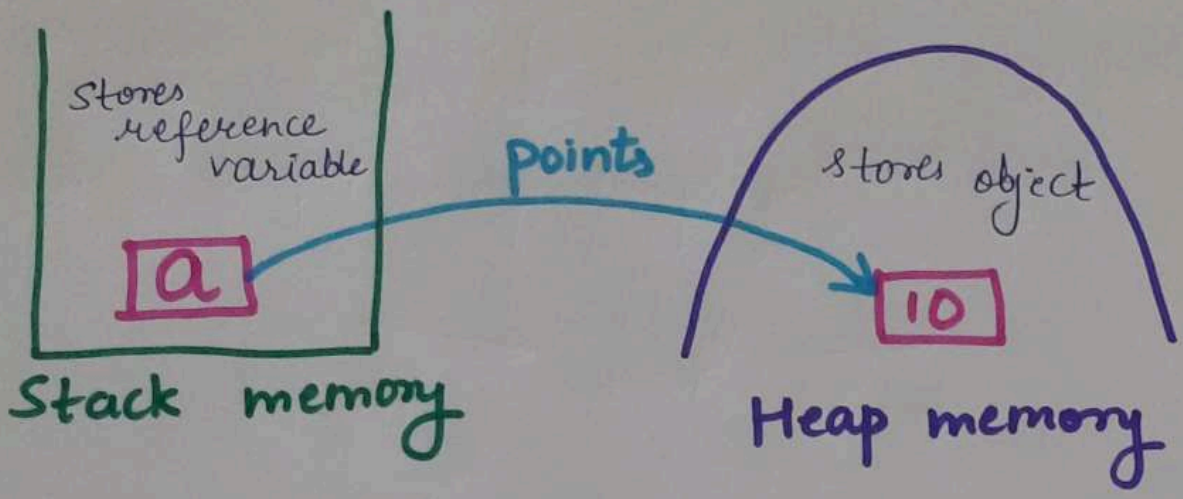
→ error might not show till programs run
→ no need to declare datatype of variables

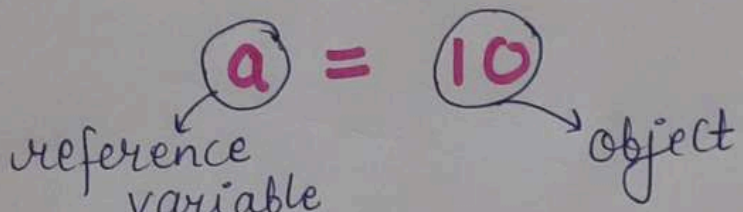
```
a = 10
```

[language by itself figures out data type]

→ Saves time in writing code but might give error at runtime.

→ Memory Management :

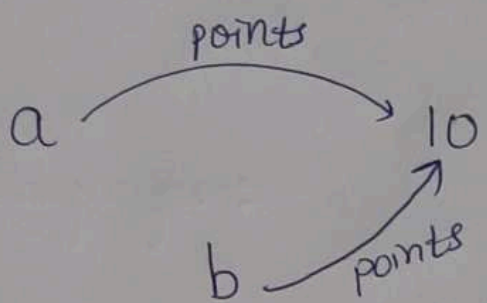




Now suppose,

a = 10

b = a



→ More than one reference variable can point towards one object.

→ If any of the reference variable changes the object then it is changed for all reference variable ~~poi~~ that points towards same object.

Now intially,

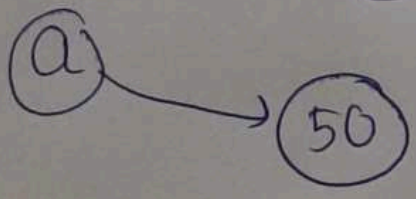
a = 10

Object with no reference variab

10 [Removed by garbage collector]

then,

a = 50



FLOW OF THE PROGRAM

Flow Chart :- Visualization of our thought process or Algorithm and represent them diagrammatically is called flow chart.

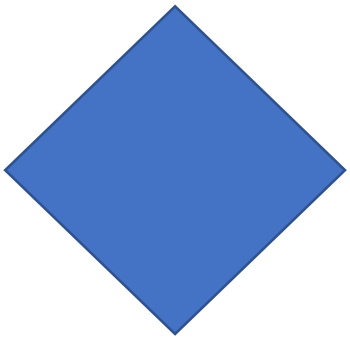
Symbols to Be used in flow chart

1. **Start / Stop** →

2. **Input / Output** →

3. **Processing** →

4. **Condition** →



5. **Flow direction of program** →

Start / Stop: - An ovel shape indicate the starting and ending points of the flow chart.

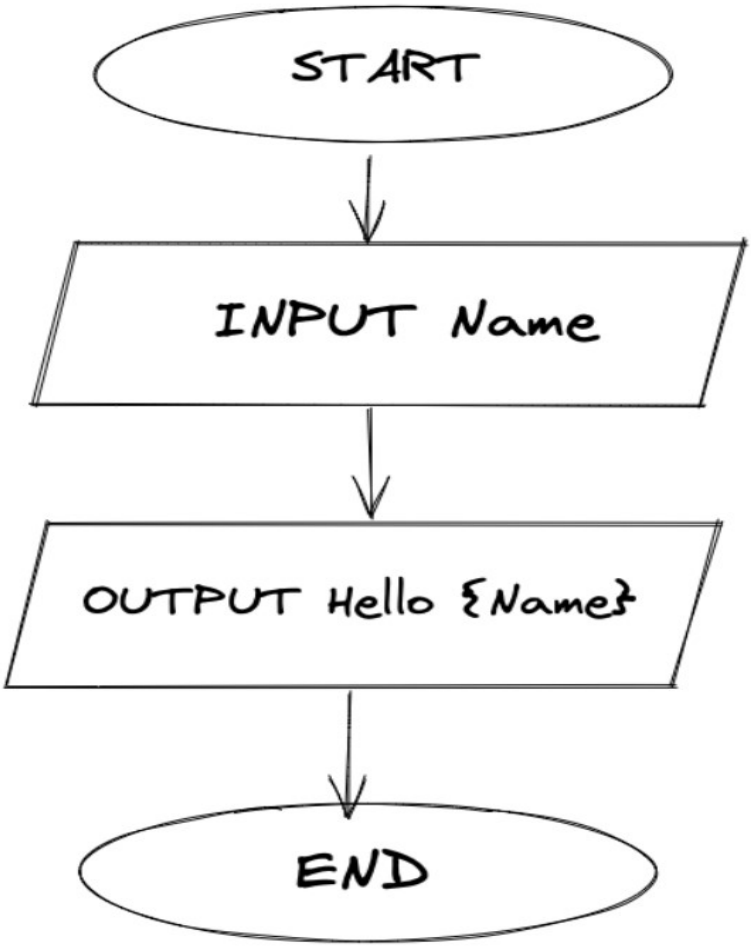
Input / Output: - A parallelogram is used to represent Input and output in flow chart

Processing: - A rectangle is used to represent process such as mathematical computation or variable assignment.

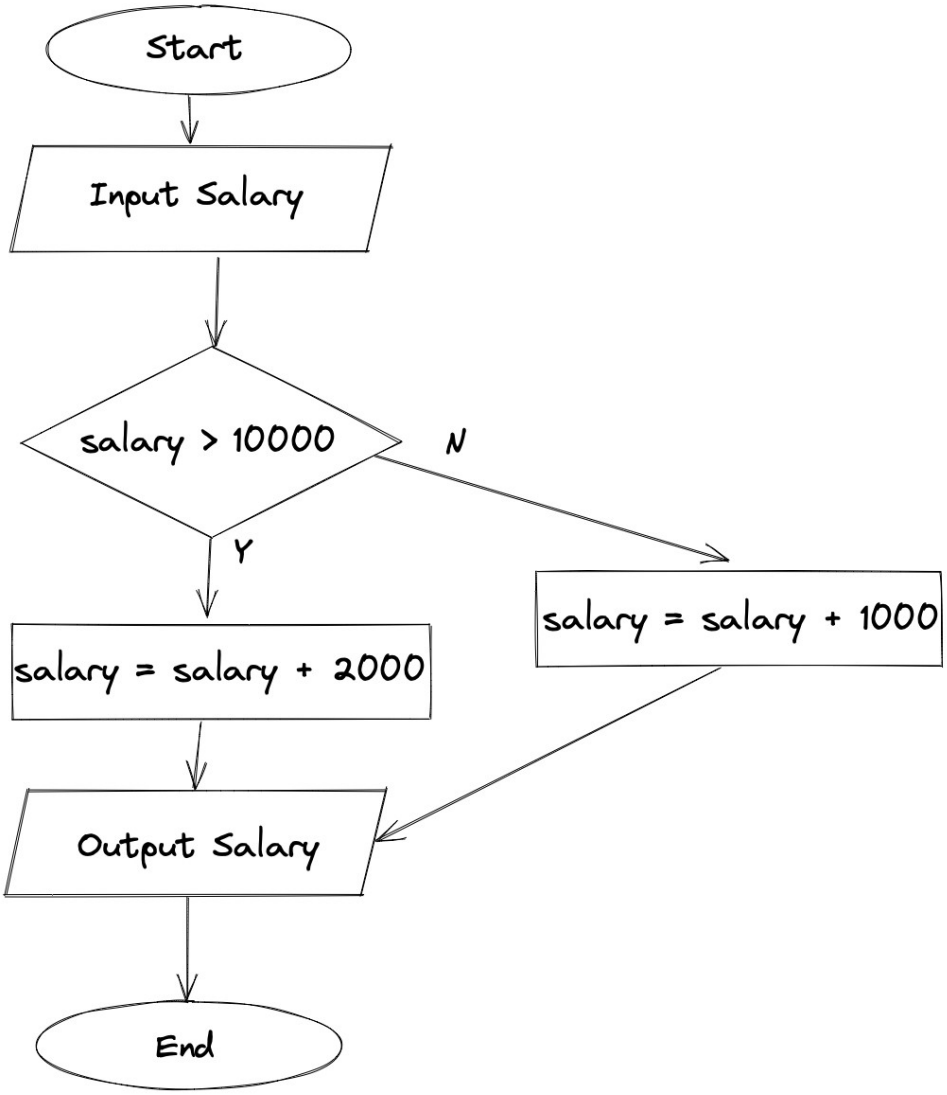
Condition: - A diamond shape is used to represent conditional statement which results in true or false (Yes or No).

Flow direction of program: - An arrow shape is used to represent flow of the program.

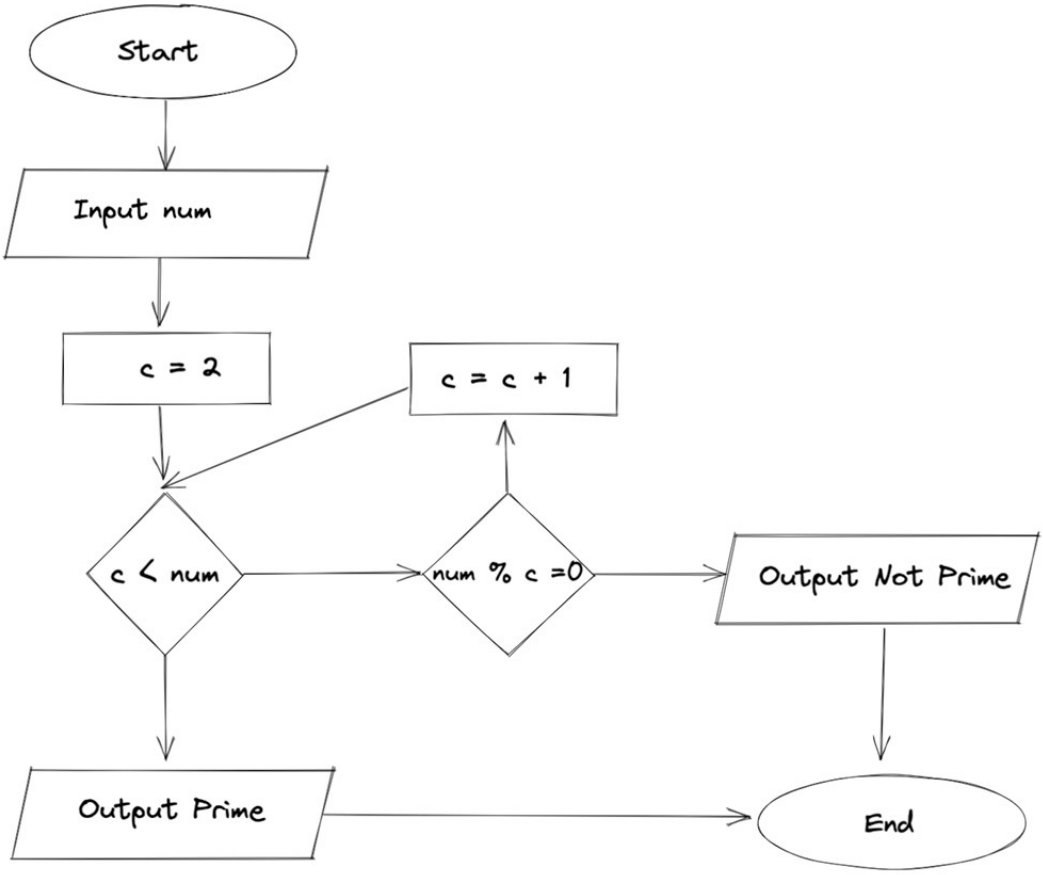
Example 1:- Take a name and output Hello name.



Example 2 :- Take input of a salary. If the salary is greater than 10,000 add bonus 2000, otherwise add bonus as 1000.



Example 3 :- Input a number and print whether it is prime or not.



Pseudocode :- It is like a rough code which represents how the algorithm of a program works.

- Pseudocode does not require syntax.

Pseudocode of Example 2

```
Start
Input Salary
if Salary > 10000 :
    Salary = Salary+2000
else :
    Salary = Salary+1000
Output Salary
exit
```

Pseudocode of Example 3

```
Start
Input num
if num ≤ 1
   print “Nither prime nor composite”
 c = 2
 while c < num
   if num % c = 0
        Output “Not Prime”
        Exit
   c = c+1
 end while
 Output “Prime”
 Exit.
```

Optimization of prime solution

Let's have a number to check it's a prime number of not

→ 36

$$\begin{bmatrix} 1 \times 36 = 36 \\ 2 \times 18 = 36 \\ 3 \times 12 = 36 \\ 4 \times 9 = 36 \end{bmatrix}$$ **…….. (i)**

$$6 \times 6 = 36$$

$$\begin{bmatrix} 9 \times 4 = 36 \\ 12 \times 3 = 36 \\ 18 \times 2 = 36 \\ 36 \times 1 = 36 \end{bmatrix}$$ **……(ii)**

In the above demonstration we have clearly seen that (i) and (ii) are repeated so, to optimize this we can ignore the (ii)

As same as this

We can check the number is prime or not by travelling form
$2\ to\ \sqrt{number}$

For example:-

- **To check 23456786543 is prime or not, we only have to travel from $2\ to\ \sqrt{23456786543}$ (i.e. 153156)**
- **To check 17 is prime or not, we do not have to travel from 2 to 17 we just have to travel from $2\ to\ \sqrt{17}$ (i.e. 4)**

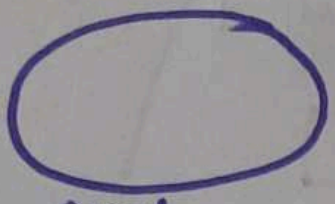
Optimized Pseudocode of Example 3

```
start
input n
if n <= 1:
        print("neither prime nor composite")

    c = 2
    while c*c <= n:
        if n % c == 0:
            output "not prime"
            exit
        c += 1
    end while
    output "prime"

exit
```

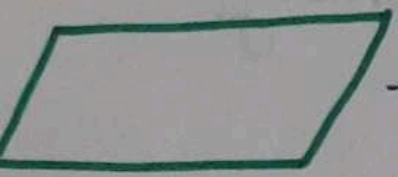
2/8/21

Flow of program

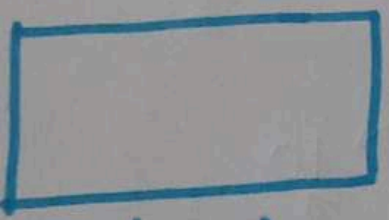
* Flow Chart Symbols :



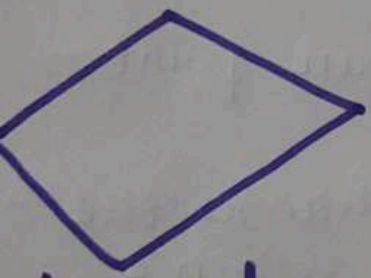
oval → Start/Stop → Represents start or end point of program.



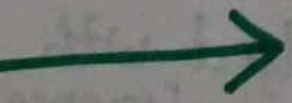
parallelogram → Input/Output → Represents the input & output



rectangle → Processing → Represents a process like addition, subtraction et

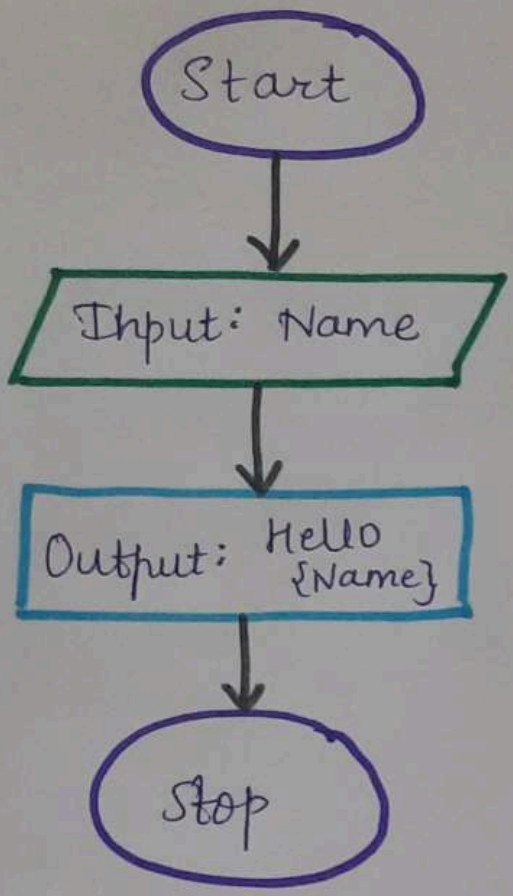


diamond → Condition → Represents for conditional statement



→ Flow direction of program → A line connector which shows what is the flow of program.

eg: Take a name & output Hello Name :



* Pseudo Code :

It is just a way to write steps which is human readable format. [It is not a code].

It is mainly meant for human reading not for machine reading.

eg: Take above example to take a name & output Hello name :

Step 1 : → Start

Step2 : → Take Input from user [name = Input ("enter na

Step3 : → Hello {Name} [output]

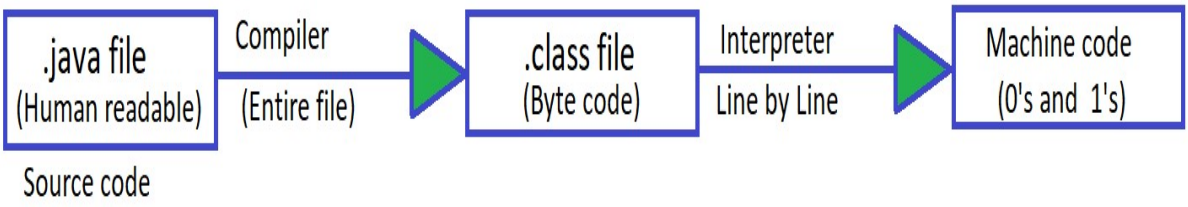
Step4 : → Stop

INTRODUCTION TO JAVA

Ques – Why do we use Programming language?

Ans – Machine only understand 0's and 1's, for humans it is very difficult to instruct computer in 0's and 1's so to avoid this issue we write our code in human readable language (Programming language).

" Java is one of the Programming Language"



- The code written in java is human readable and it is saved using extension **.java**
- This code is known as source code

Java Compiler:-

- Java compiler converts the source code into byte code which have the extension **.class**
- This byte code not directly run on system
- We need JVM to run this
- Reason why java is platform independent

Java Interpreter

- Converts byte code to machine code i.e. 0's and 1's
- It translate the byte code line by line to machine code

More About Platform Independent

→ It means that byte code can run on all operating system

→ We need to convert source code to machine code so computer can understand it.

→ Compiler helps in doing this by turning it into executable code.

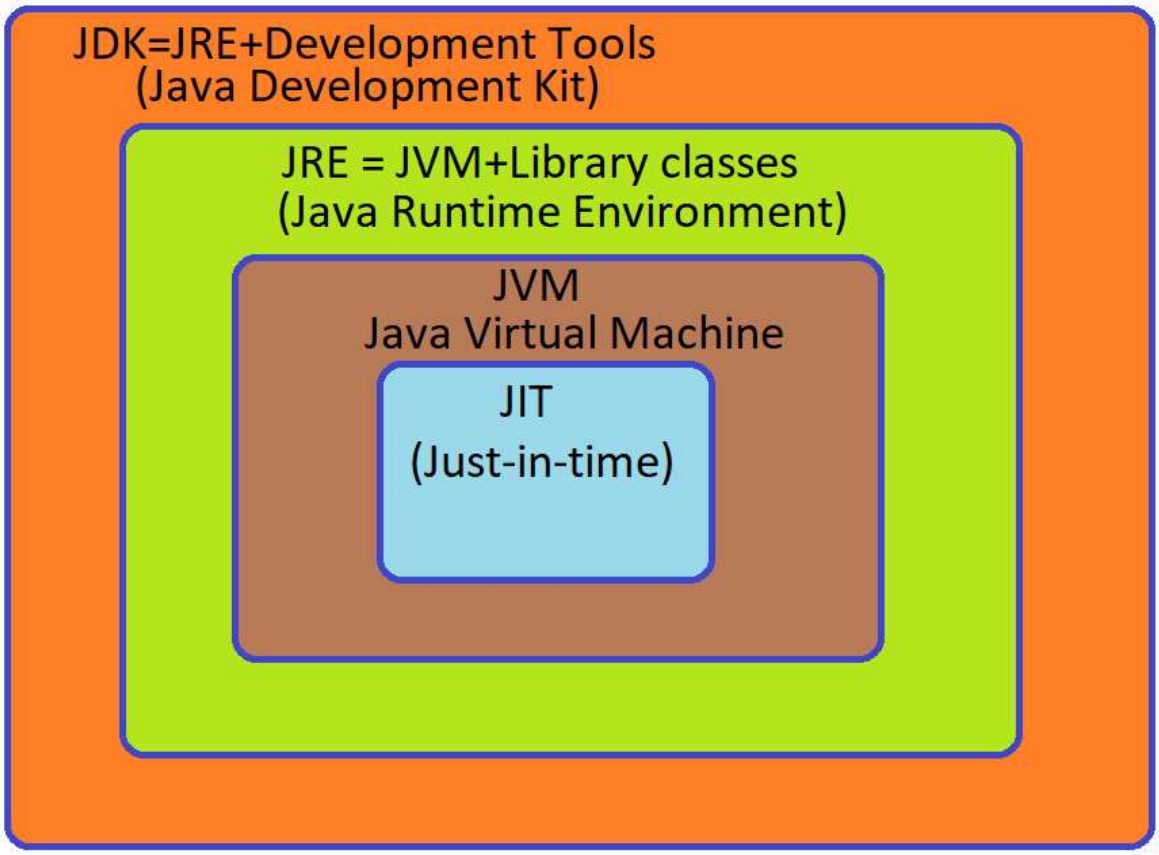
→ This executable code is a set of instruction s for the computer

→ After compiling C/C++ code we get .exe file which is platform dependent.

→ In java we get byte code, JVM converts this to machine code.

→ Java is platform independent but JVM is platform dependent.

Architecture of Java



JDK

- Provide Environment to develop and run the java program.
- It is a package that includes :-
  1. **Development tools :-** To provide an environment to run your program.
  2. **JRE :-** To Execute your program.
  3. **A compiler :-** javac
  4. **Archiver :-** Jar
  5. **Docs generator :-** Javadoc
  6. **Interpreter/loader**

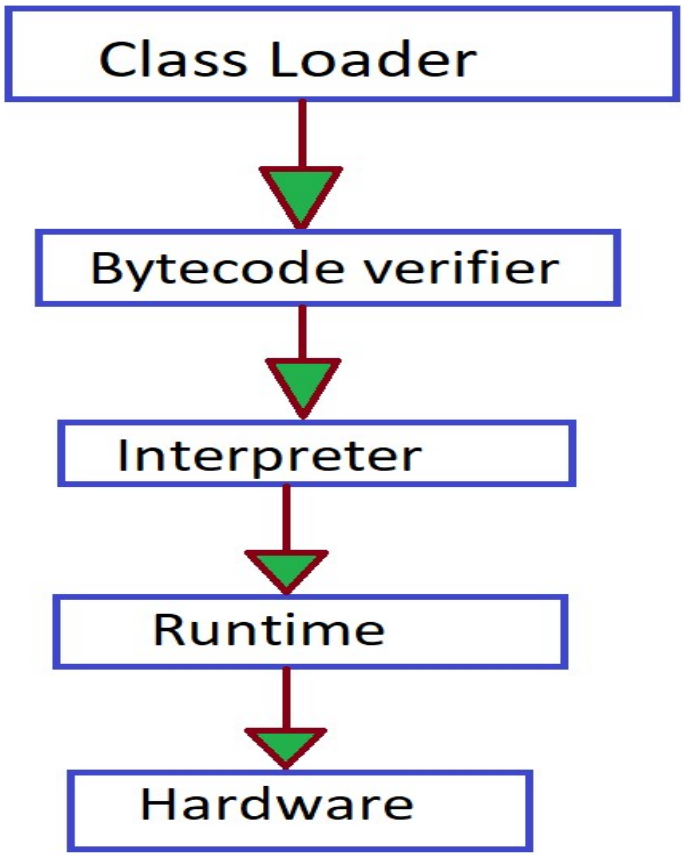
JRE

- It is an installation package that provides environment to only run the program.
- It consist of :-
  1. **Deployment technology**
  2. **User interface toolkit**
  3. **Integration libraries**
  4. **Base libraries**
  5. **JVM** :- Java virtual Machine

Compile Time:-

- After we get the .class file the next thing happen at runtime :
  1. Class loader loads all classes needed to execute the program.
  2. JVM sends code to bytecode verifier to check the format of code.

Runtime :-



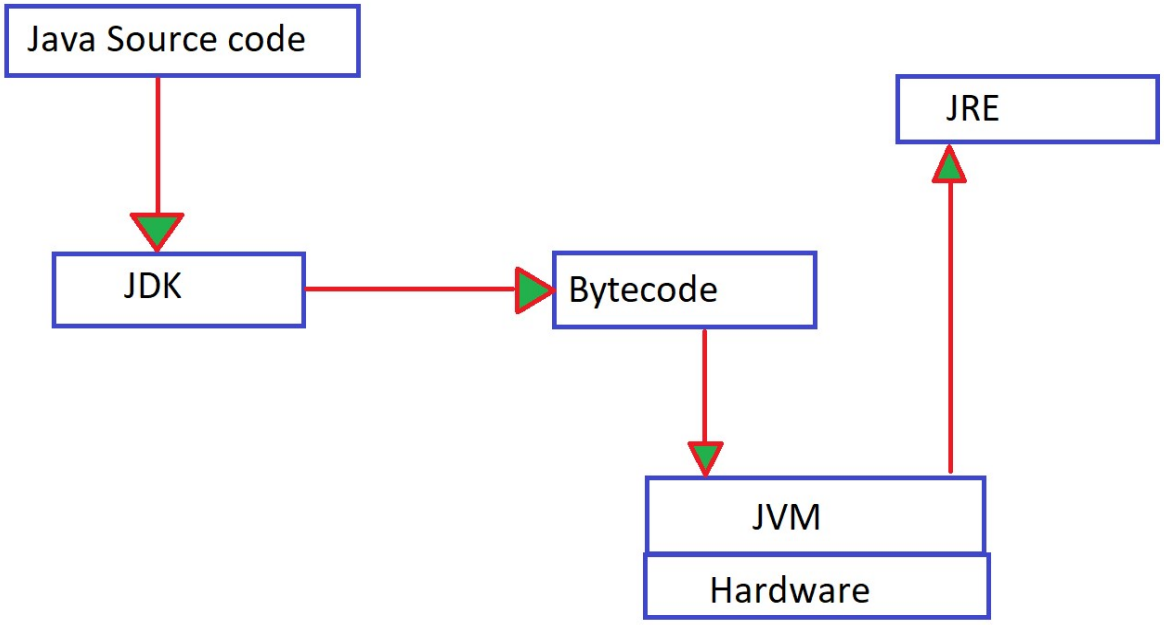
(How JVM Works) class Loader

- ***Loading***
  → Read .class file and generate binary data.
  → an Object of this class is created in heap
- ***Linking***
  → JVM verifies the .class file
  → allocates memory for class variables and default values
  → replace symbolic references from the type with direct reference.
- ***Initialization***
  → All static variables are assigned with their values defined in the code and static bock.
  → JVM contains the stack and heap memory locations.

JVM Execution

- ***Interpreter***
  → Line by line execution
  → When one method is called many times it will interpret again and again
- ***JIT***
  → Those methods that are repeated, JIT provides direct machine code so that interpretation is not required.
  → Makes execution Faster.
  → **Garbage collector**

Working of Java Architecture



Tools required to run java on machine

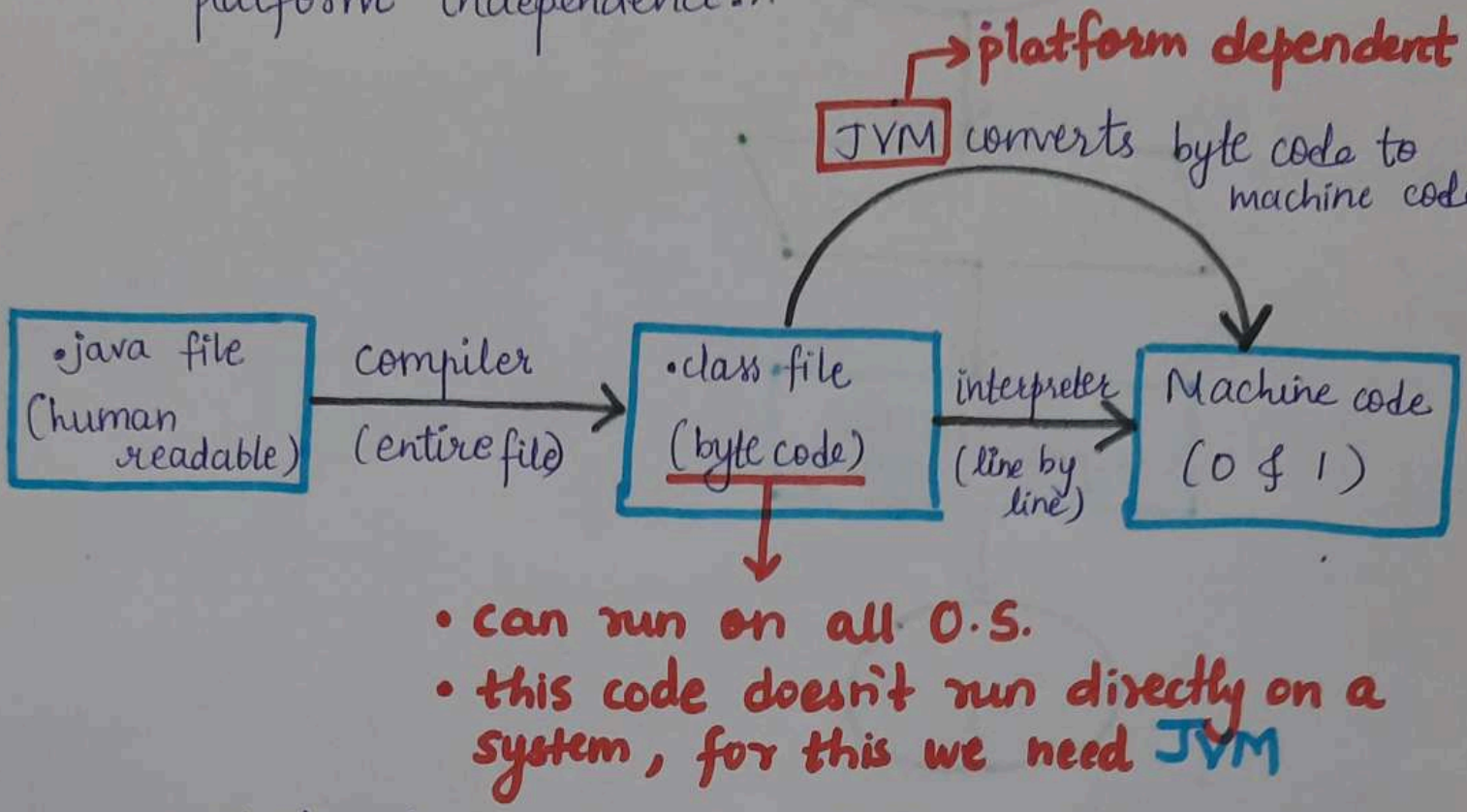
1. **JDK** https://www.oracle.com/in/java/technologies/javase-downloads.html

2. **InteliJ**
   a) For windows :- https://www.jetbrains.com/idea/download/#section=windows
   b) For macOS :- https://www.jetbrains.com/idea/download/#section=mac
   c) For Linux :- https://www.jetbrains.com/idea/download/#section=linux

3/8/21

Introduction to Java ♥

★ How Java code executes and more information about platform Independence...



- can run on all O.S.
- this code doesn't run directly on a system, for this we need JVM

★ Therefore, Java is platform independent ★

⇒ We can provide this byte code to any system means we can compile the java code on any system.

⇒ But JVM is platform dependent means for every O.S. the executable file that we get, it has step by step set of instruction dependent on platform.

✯ JDK vs JRE vs JVM vs JIT

JDK [Java Development Kit]

↳ provides environment to develop & run Java program

JRE [Java Runtime Environment]

↳ provides environment to only run the program

- → deployment technologies
- → user interface toolkit
- → integration libraries
- → base libraries

etc.

JVM [Java Virtual Machine]

JIT [Just-in-time]

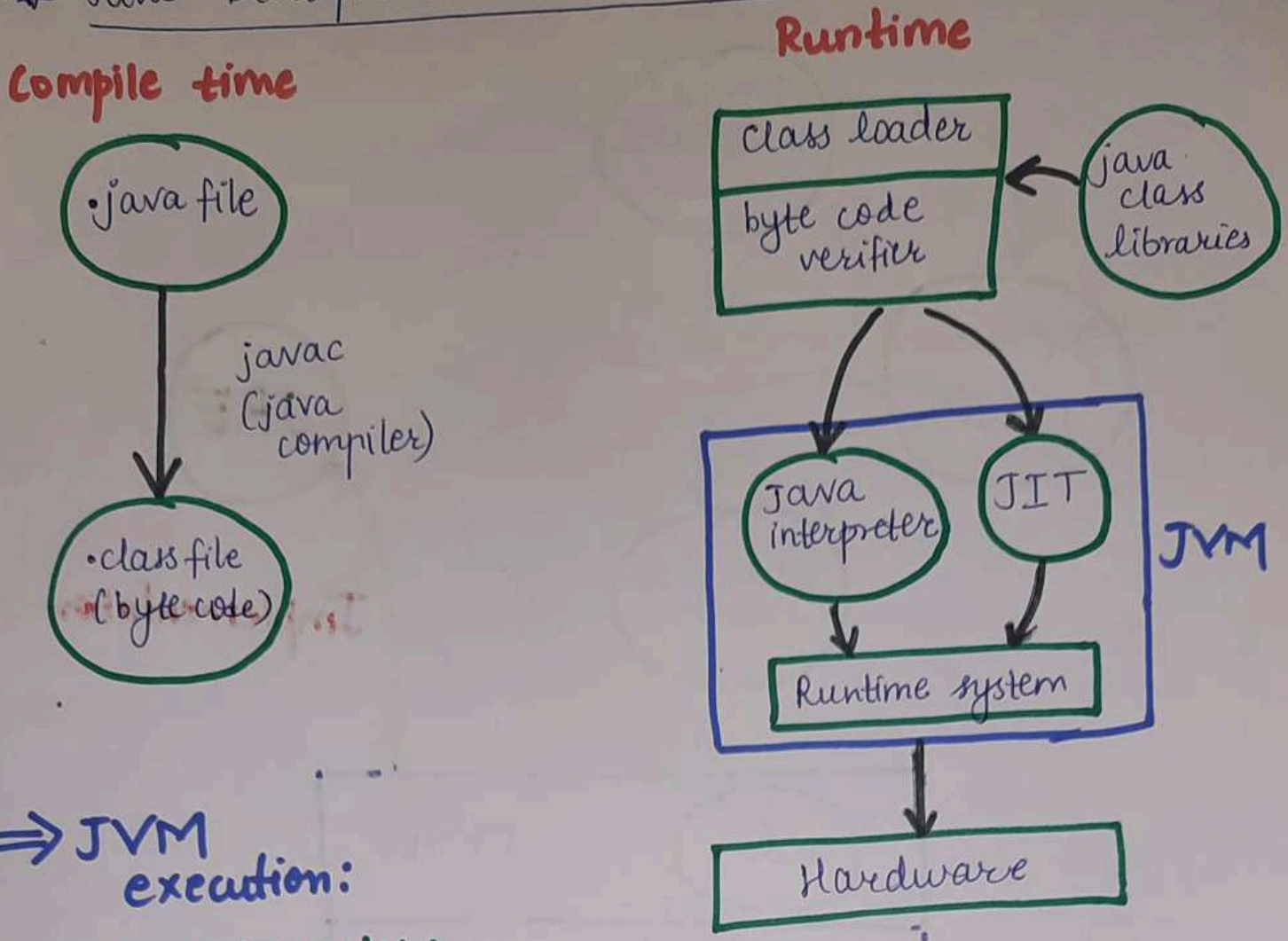
- → Java Interpreter
- → Garbage collector

etc.

- → development tools
- → javac → Java compiler
- → archiver → jar
- → docs generator ↳ javadoc
- → interpreter/loader

etc.

✱ Java Development and Runtime Environment



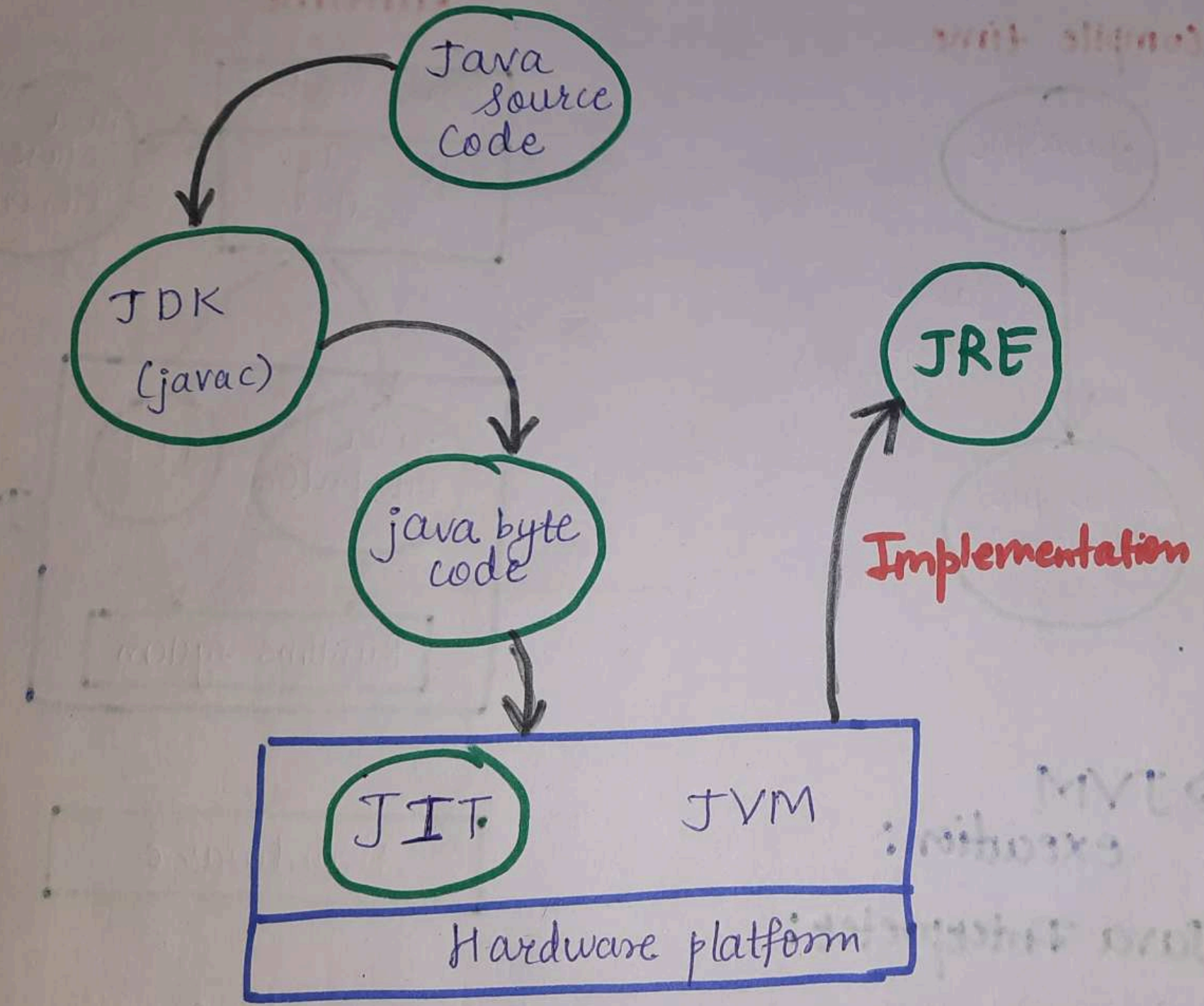
⇒ JVM execution:

- **Java Interpreter:**
  - → line by line execution
  - → when one method is called many times, it will interpret again & again
- **JIT:**
  - → methods that are repeated, JIT provides direct machine code so re-interpretation is not required
  - → makes execution faster
- **Garbage Collector**

✱ Class loader:

- **Loading**
  - → reads byte code file & generates binary data
  - → an object of this class is created in heap
- **Linking**
  - → JVM verifies .class file
  - → allocates memory for class variables & default values
  - → replace symbolic references from the type with direct references
- **Initialization**
  - → all static variable are assigned with their values defined in the code & static block.

Summary:


Java Source Code
JDK (javac)
java byte code
JRE
Implementation
JIT
JVM
Hardware platform

FIRST JAVA PROGRAM

Structure of Java File

"Source code that we write will be saved using extension .java"

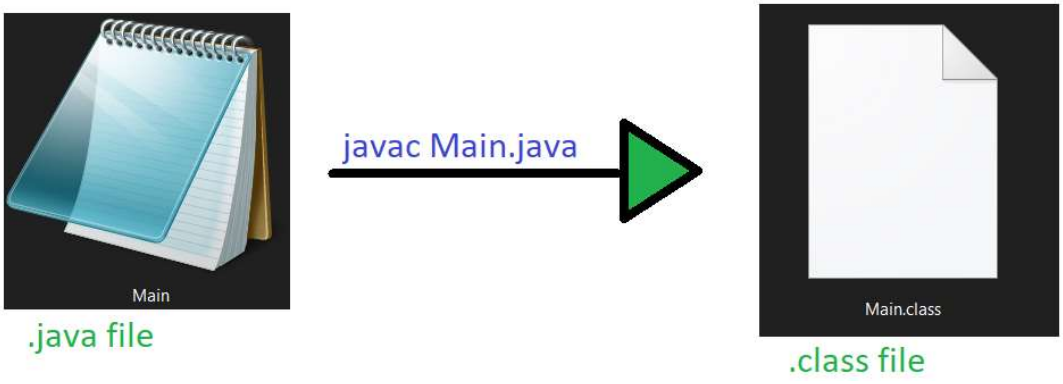
- Every thing written in .java file must be in classes or we can say that every file having .java extension is a class
- A class with same name as file name must be present in .java file.

First alphabet of class name can be in upper case. It is the naming convention of class name. however, it is not compulsory to do so.

- Class which is having same name as file must be public class
- A main function/method must be present in this public class, main is a function from where the program starts.

Converting .java to .class

- Using javac compiler we can convert .java file to .class
  Command to convert .java to .class
  Javac and .java file name
  Let the name of .java file is Main, so the command to convert .java to .class is
  Javac Main.java
- Above command create a . class file (Main.class) which contains Bytecode.



Running the program

By using java and name of file we can run the program.

- Command > **java Main**

Hello world program

```
public class Main{
    public static void main(String [] args){
        System.out.println("Hello World");
    }
}
```

1. **public (in first line):-** public is an access modifier which allows to access the class from anywhere.
2. **class :-** It is a name group of properties and functions
3. **Main :-** It is just the name of class as same as the name of file.
4. **public (in second line) :-** It is used to allow the program to use main function from anywhere.
5. **static :-** It is a keyword which helps the main method to run without using objects.
6. **void :-** It is a keyword used when we do not want to return anything from a method/function
7. **main :-**It is the name of method.
8. **String [] args :-** It is a command line argument of string type array.
9. **System:-** It is a final class defined in java.lang package.
10. **out :-** It is a variable of PrintStream type which is public and static member field of the System class.
11. **println :-** It is a method of PrintStream class, It prints the argumants passed to it and adds a new line. **print** can aalso be used here but it prints only arguments passed to it. It do not adds a new line.

What is package ?

→ It is just a folder in which java files lies.

→ It is used to provide some rules and stuff to our programs.

Primitive data types

- Primitives data types are those data types which is not breakable.

  Ex:-
  String is not a primitive data type so we can break this data type into char
  i.e., String “Kunal” can be divided into
  ‘K’ ‘u’ ‘n’ ‘a’ ‘l’
  But primitives data type are not breakable.
  We cannot break a char ,int etc.

- List of primitive data types in java are :-

Data types	Description	Example
int	int is used to store numeric digits	int i = 26;
char	char is used to store character	char c = ‘A’;
float	flot is used to store floating point numbers	float f = 98.67f;
double	double is used to store larger decimal numbers	double d = 45676.58975 ;
long	long is used to store numeric digits which is not able to stored in int	long l = 15876954832558315l;
boolean	It only stores store t values i.e., true or false.	boolean b = false;

In float and long we have used f and l, it denotes that the number in the variable is float or long type, if we do not use this java consider float value as double and long value as int.

- Literals :- It is a synthetic representation of boolean, character, string, and numeric data.
  Ex:- int a = 10;
  Here 10 is called literal.
- Identifiers:- name of variable, methods, class, packages, etc. are known as identifiers.
  Ex:- int a = 10;
  Here a is Identifier.

Comments in Java

Comments are something which is written in source code but ignored by compiler.

- Two types of Comment
  1. Single line comment :- used to comment down a single line (// is used for it.)
  2. Multi line comment :- used to comment down multiple lines (/* */ is used for it)

Inputs in Java

We have Scanner class available in java.util package to take input

To use this class we have to

1. Import java.util package in our file.
2. Create object of the scanner class
3. Use that object to take input from the keyboard.

Syntax :-

```
import java.util.Scanner;
public class Main{
    public static void main(String [] args){
        Scanner input = new Scanner(System.in);

    }
}
```

1. **Scanner :-** It is a class required to take input, it is present in java.util package.
2. **input :-** It is an object that we are creating to take input.
3. **new :-** It is a keyword used to create an object in java.
4. **System.in :- System** is a class and **in** is a variable that denotes we are taking input from standard input stream (i.e. Keyboard).

int Input :- nextInt() is a function used to take input of int.

Syntax:-

```
Scanner input = new Scanner(System.in);
        int rollno = input.nextInt();
```

float Input :- nextFnt() is a function used to take input of int.

Syntax:-

```
Scanner input = new Scanner(System.in);
        float marks = input.nextFloat();
```

String Input :- Two ways to take string input

1. Using next() Method :- It will take one word input till a space occurs

   Syntax:-

```
Scanner input = new Scanner(System.in);
        String s1 = input.next();
```

   Input :- Hey kunal

   Output :- Hey

2. Using nextLine() Method :- It will take all string input including space.
   Syntax:-

```
Scanner input = new Scanner(System.in);
        String s2 = input.nextLine();
```

Sum of two numbers

```java
import java.util.Scanner;

public class Sum {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter first number");
        int num1 = input.nextInt();
        System.out.print("Enter second number");
        int num2 = input.nextInt();
        int sum = num1+num2;
        System.out.println("Sum = "+sum);

    }
}
```

Output

```
Enter first number70
Enter second number80
Sum = 150
```

Type conversion

When one type of data is assigned to another type of variable an automatic type conversion will take place under some condition

Conditions :-

1. Two types should be compatible.
2. Destination type should be greater then the source type.

Type Casting

When we convert one type of data to another type is known as type casting

Ex:- int num = (int) (67.564f)

Automatic type promotion in expressions.

While evaluating expressions the intermediate value may exceed the range of operands and hence the expression value will be promoted.

There are some condition for type promotion:-

1. Java automatically promotes each byte, short or char operand to int when evaluating an expression.
2. If one operand is a long , float or double the whole expression is promoted to long , float or double respectively.

Ex:-

```
byte a = 40;
byte b = 50;
byte c = 100;
int  d = (a*b)/c;
System.out.println(d);
```

Here when a*b occurred it became 2000 which is out of the range of byte so here byte is automatically promoted to int type.

Example for thorough review concept.

```
public class TypePromotion {
    public static void main(String[] args) {
        byte b = 42;
        char c = 'a';
        short s = 1024;
        int i = 50000;
        float f = 5.67f;
        double d = 0.1234;
        double result = (f*b)+(i/c)-(d*s);
        System.out.println((f*b)+" "+(i/c)+" "+" "+(d*s));
        System.out.println(result);

    }
}
```

Output

```
238.14 515   126.3616
626.7784146484375
```

Prime number program.

```java
import java.util.Scanner;
public class Prime {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Please enter a number");
        int n = in.nextInt();
        if(n<=1){
            System.out.println("Nither prime nor composite");
            return;
        }
        int c=2;
        if(n==4){
            System.out.println("Not Prime");
        }
        else{
            while(c*c<n){
                if (n%c==0){
                    System.out.println("Not Prime");
                    return;
                }
                c=c+1;
            }
            if(c*c>n){
                System.out.println("Prime");
            }
        }
    }
}
```

Output :-

1.
```
Please enter a number
17
Prime
```

2.
```
Please enter a number
1
Nither prime nor composite
```

3.
```
Please enter a number
6
Not Prime
```

Example of if statement.

Statement inside if statement only executes when condition given in if is true.

```
public class ifstatement {
    public static void main(String[] args) {
        int a = 10;
        if (a == 10){
            System.out.println("Hello");
        }
    }
}
```

output

```
Hello
```

Example of while loop.

Statement in while loop run till condition in while loop become false

```
public class whileloop {
     public static void main(String[] args) {
            int count = 1;
           while (count != 5) {
            System.out.println("count");
            count++;
         }
     }
 }
```

Output

```
count
count
count
count
```

Example of for loop.

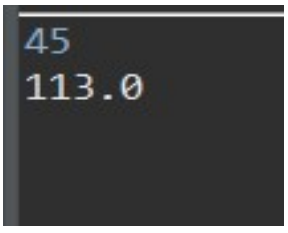
```
public class forloop {
    public static void main(String[] args) {
        for (int count=1;count!=5;count++){
            System.out.println(count);
        }
    }
}
```

Output

Celsius to Fahrenheit program.

```
import java.util.Scanner;

public class CelsiusToFahrenheit {
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        float tempC = in.nextFloat();
        float tempF = (tempC*9/5)+32;
        System.out.println(tempF);
    }
}
```

Output



3/8/21

First Java Program - Input/Output, Debugging & Datatypes

File name: **Demo.java** Class Name: **Demo**

Its good practice to use initial character as capital (you can use small also)

public → this keyword means, it is used so that we can access the class from anywhere.

functions → Collection of code, that we can use again & again. Functions are also known as methods.

void → The void keyword specifies that a method should not have a return value.

String[] args → means an array of sequence of characters ("strings") that are passed to the main function.

array

- After compiling, .class file is always saved in current location where you are in.
- If you want to change the location, use **-d** (destinato option while compiling and specify the path.

javac -d <path> Demo.java

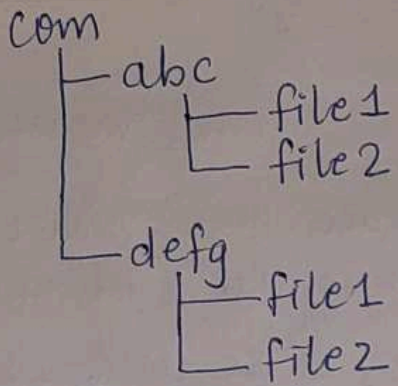
- **echo $PATH** → every command looks for this location before executing.

[environment variables]

- class name & file name should be same, but if we don't want to make class name as file name then it should not be public.

for eg → **class Divide**

- package com.abc OR package com.defg



- `System.out.println("Hello");` → this means print the output on Standard output stream (here, terminal)
  - `System` → class
  - `out` → var
  - `println` → fn/method

println → adds new line
print → does not add new line.

- `Scanner input = new Scanner(System.in);`
  - `Scanner` → class that allows us to take input
  - `new` → creating object
  - `System.in` → take input from standard input (here, keyboard)

Primitive → means any data type that cannot be broke further.
integer, character etc. are primitive datatype.

⇒
```
int rollno = 64;  → 4 bytes
char letter = 'r';
float marks = 98.67f;  → 4 bytes
double largeDecimalNumbers = 456789.12345;  → 8 bytes
long largeIntegers = 12345678l0L;  → 8 bytes
boolean check = true;
```

- String is written in double quotes whereas while specifying char we write it in single quotes.
- All decimal values that we use are by default of double datatype, therefore if we want to store in float we have to use "f", same for int & long.

```
float marks = 7.2f
(by default) double largeDecimalNumbers = 456789101.12345

int rollno = 64; (by default)
long LargeInteger = 1234567891011L;
```

- Integer → Wrapper class → provides additional functionalities
  - converts primitive datatype to object.
- Comment → the lines that we comment are ignored by Java and will not be executed.
  Comment in Java → //

→ `int a = 10` → literal

a → identifier

- Literals : Java literals are syntactic representations of boolean, character, numeric or string data.
  here, 10 is an integer literal.
- Identifiers : Identifiers are the names of variables, methods, classes, packages & interfaces.

- **int a = 234_000_000;**
  ↳ the value of a will be 234000000, underscore will be ignored.
- 564.12345678 $\xrightarrow{\text{rounds off}}$ 564.12345
  If we give float very big, than it rounds off the value which gives floating point error.

⟹ Type Casting & Type Conversion:

• Widening or Automatic Type Conversion:

→ Two datatypes are automatically converted.
→ This happens when we assign value of smaller datatype to bigger datatype & two datatype must be compatible.

byte → short → int → long → float → double

eg→ `int i = 100;` → 100
`long l = i;` → 100
`float f = l;` → 100.0

• Narrowing or Explicit Conversion:

→ This happens we want to assign a value of larger data type to a smaller data type we perform explicit type casting or narrowing.

double → float → long → int → short → byte

eg→ `double d = 100.04;` → 100.04
`long l = (long) d;` → 100
`int i = (int) l;` → 100

• Automatic Type Promotion in Expressions :

→ While evaluating expressions, the intermediate value may exceed the range of operands & hence the expression value will be promoted.

→ Some conditions of type promotion are:

1. Java automatically promotes each byte, short, char to int when evaluating an expression
2. Long, float or double the whole expression is promoted to long, ~~whole~~ float or double.

eg: After solving expression :

```
(f * b) + (i / c) - (d * s);
```

we get → float + int − double = double

converted to biggest one

• Explicit type casting in expressions :

→ If we want to store large value into small data type

eg:
```
byte b = 50;
b = (byte)(b * 2);
```
→ type casting int to byte.

• If-else syntax in Java

```
if (condition) {
  // block of code
} else {
  // block of code
}
```

• For loop syntax

```
for (statement1 ; statement2 ; statement3) {
  // code block
}
```