OOP Revsion OOSD -> Object Oriented Software Development

1. OOA -> Analysis of all the entities
2. OOD
3. OOP-> java

```
class Employee{
int id
String name
salaray
}
```

1. Major Pillars
   - Abstraction
   - Encapuslation
   - Modularity
   - Hirerachy
2. Minor Pillars
   - Typing/polymprphism
   - concurrency
   - Persistance

Abstraction -> Encapsulation
Function call -> Define a Function
Creating Object -> Defining class

Modularity -> Divide the code into multiple functions/classes/files
Hirerachy -> Relationship between the entities/classes represents hirerachy

has-a

```
class Engine{

}
```

```
class Car{
Engine e;
}
```

is-a

```
class Base{

}
```

```
class Derived : public Base{

}
```

Minor Pillars
1. Polymorphism
   - Compile
     - fun overloading
   - Runtime
     - fun overriding

```
Square *sptr;
Rectangle *rptr;
Circle *cptr;
```

```
Shape *sh_ptr;
```
- new Square() / new Rectangle() / new Circle()

2. Concurrency

3. Persistance -> File / Database

Functional Programming

Procedure Oriented
C

Procedure Oriented / OOP
CPP

OOP / Functional
java

1. Java Card
   - Smart Cards
2. Java ME
   - Phones, Printers,etc...
3. Java SE
   - Desktop Applications
4. Java EE
   - Web Applications

Installations
STS-4 -> Major bug
STS -> 3.9.18

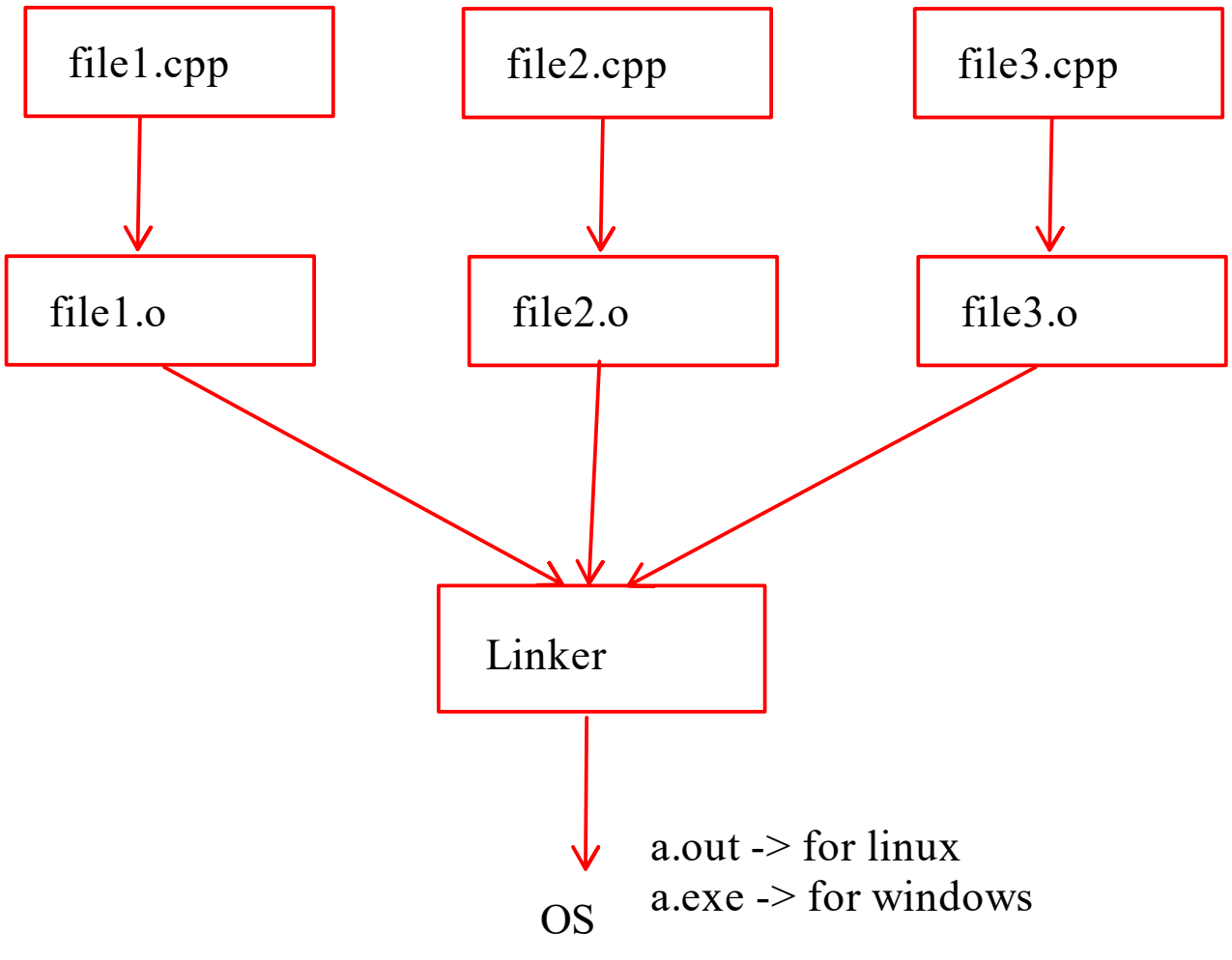
JDK-> Java Development kit

JDK -> Java Tools + docs + JRE(Java RunTime Environment)

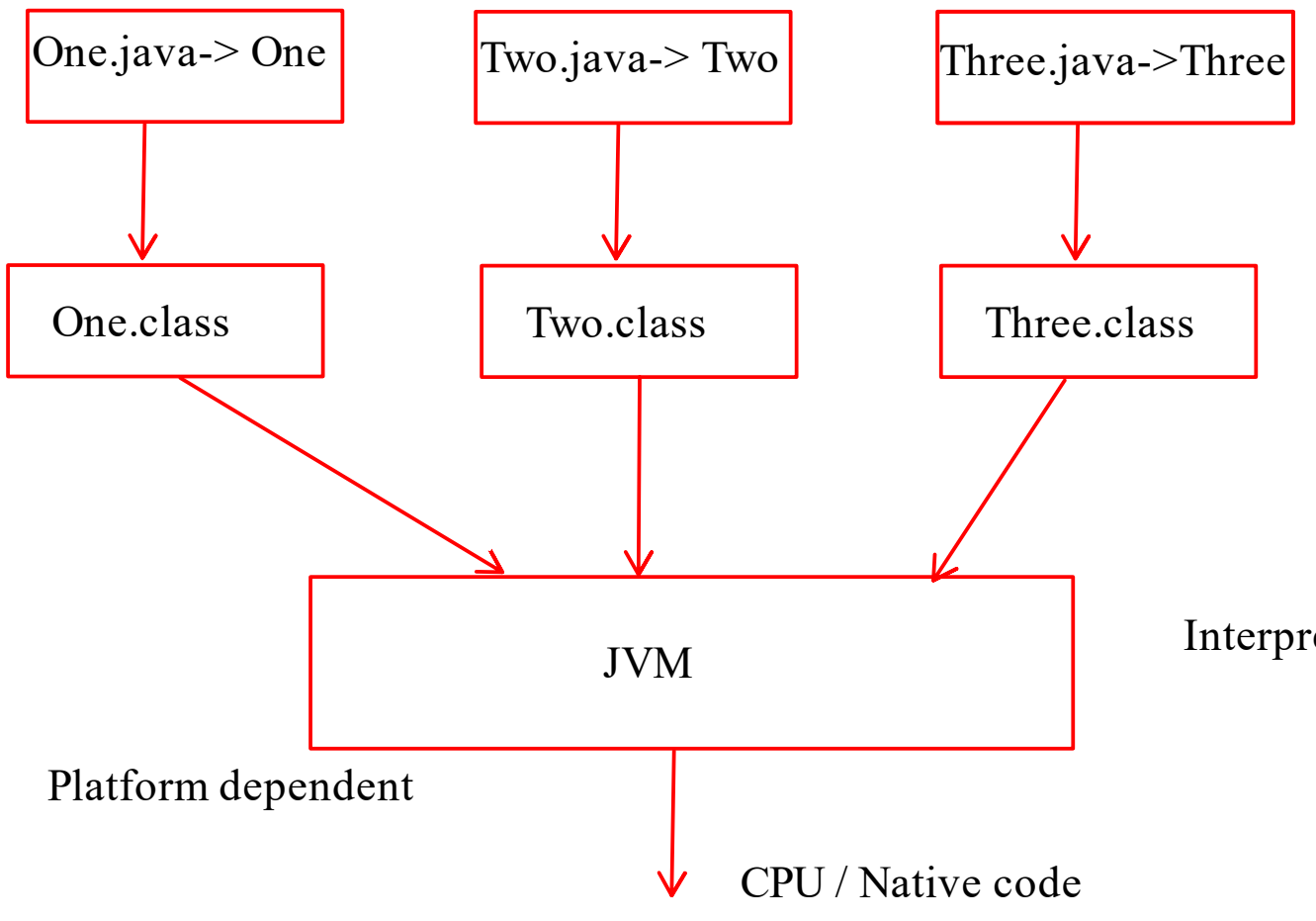
JRE -> JVM(Java Virtual Machine) + rt.jar(core libraries)(jmods)

Developer
JDK

Client
JRE + java tool



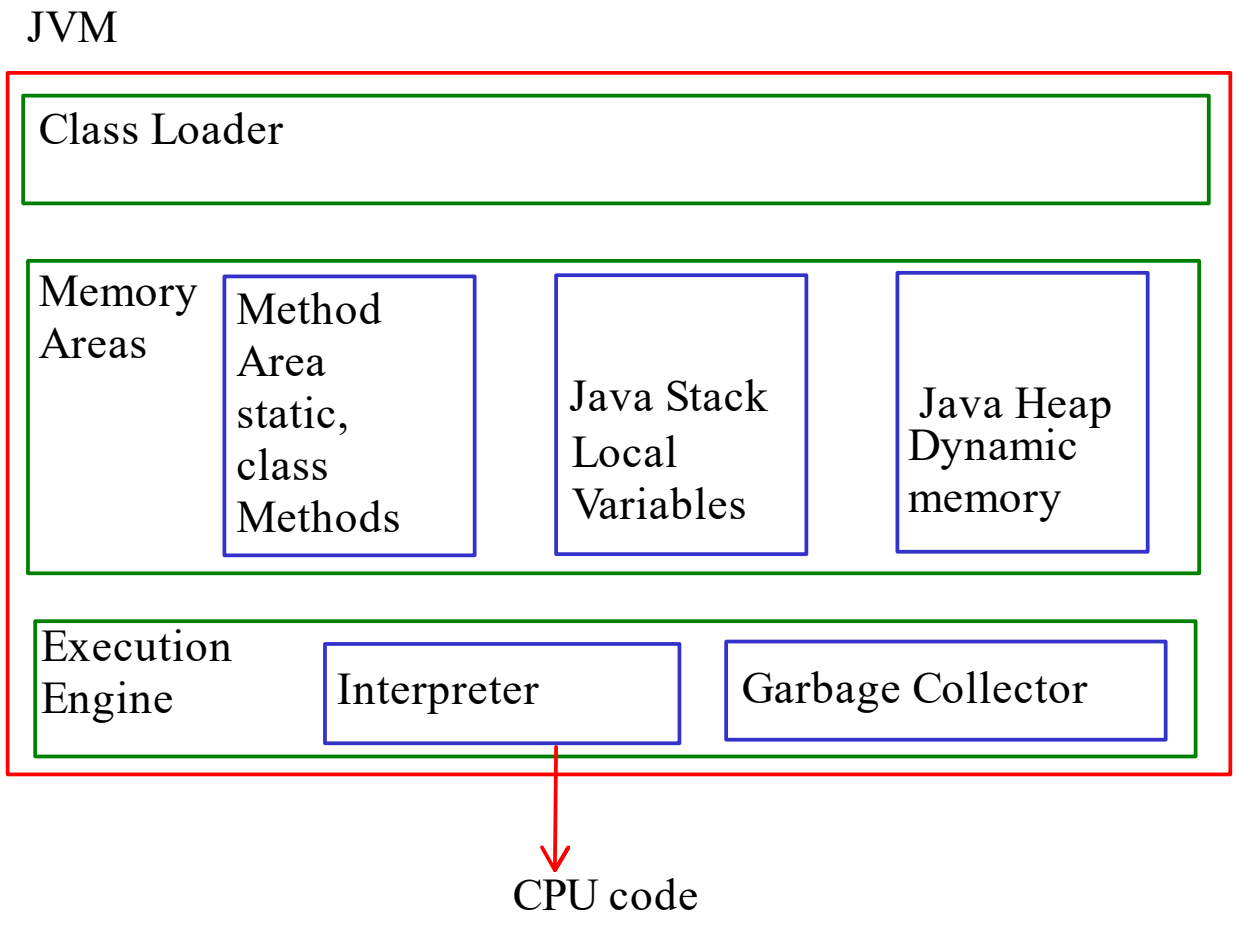
g++ file1.cpp



javac

ByteCode -> Intermediate code
understandable by JVM

Interpreating these .class files is the job of JVM




```
class{
//datamembers
fields

//functions();
Methods();
}
```

CPP->
Stack
Heap
Data
Text/code

JVM Architecture will see
Later.
2 more memory areas
- PC Register
- Native Method Stack

class {
Fields

Methods
}

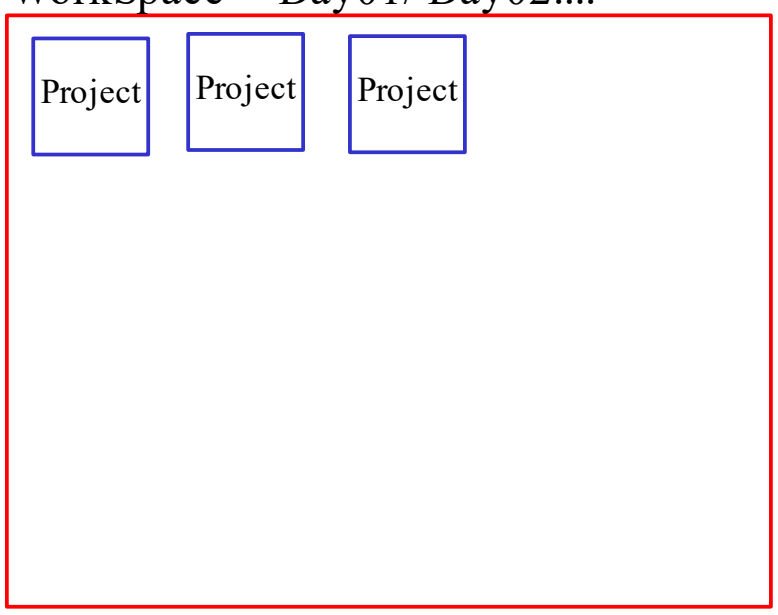
- Fields and Methods are members of the class
- class members can be static or non static
- static -> accessed using classname
- nonstatic -> on classs Object

Progarm.java
java Program.java
Program.class

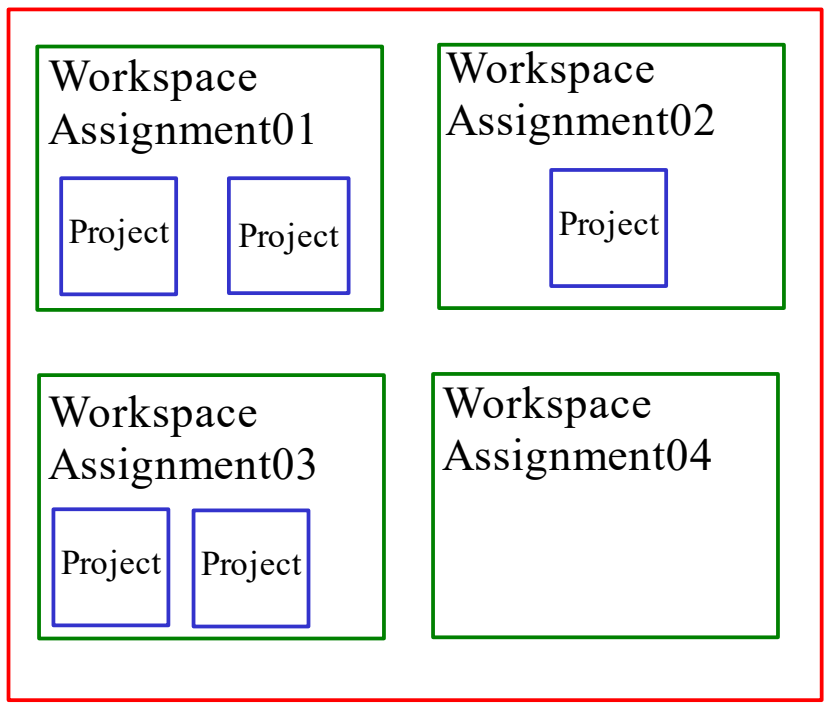
java Program

Program.main();

WorkSpace-> Day01/ Day02....



your_git_repo (java)




```
class Test{
	int field1=10;
	static int field2=20;
}

// CPP
main(){
cout<<"Value of field2 = "<<Test::field2;

Test t1;
cout<<"Value of field1 = "<<t1.field1;
}
```

```
java
main(){
sysout(Test. field2);

Test t1 = new Test();
sysout(t1.field1);
}
```

JVM -> main()

//Windows
SET CLASSPATH

// linux
echo $CLASSPATH

to set classpath in linux
export CLASSPATH=../bin

- below command should be given from src dir
javac -d ..\bin .\Program.java

- set classpath till bin dir
SET CLASSPATH=..\bin

- execute the code
java Program

- As per java language specification
1. Name of public class and name of .java file should be same
2. In single .java file we cannot declare multiple public classes

1. Scanner -> mostly used
2. Console

cout -> out
cin -> in

```
Employee *eptr = NULL;
eptr->accept();
```

CCEE -> System belongs to
1. java.lang
2. java.util
3. java.sql
4. java.io

Class, Object, Reference

Class
- It is a Logical Entity
- Blueprint of an Object

Object
- It is a physical entity
- It is called as instance of a class

Reference
- It is a varaiable that stores the object

```
main(){
int num1; // variable
static int num2;// static local variables are not allowed

Employee e; // variable -> reference
e = new Employee();
}

class Test{
int num1; // Field
static int num2; // static field
}
```

```
Employee e; // stack
Employee *eptr;
eptr = new Employee()
delete eptr;
```

from cpp-> memory areas
stack
heap
data
text/code

```
int/void function(int n1, int n2....){

    return something;
}
```

```
method();

int val = 10;
double = val;
```

11-> java 1.8