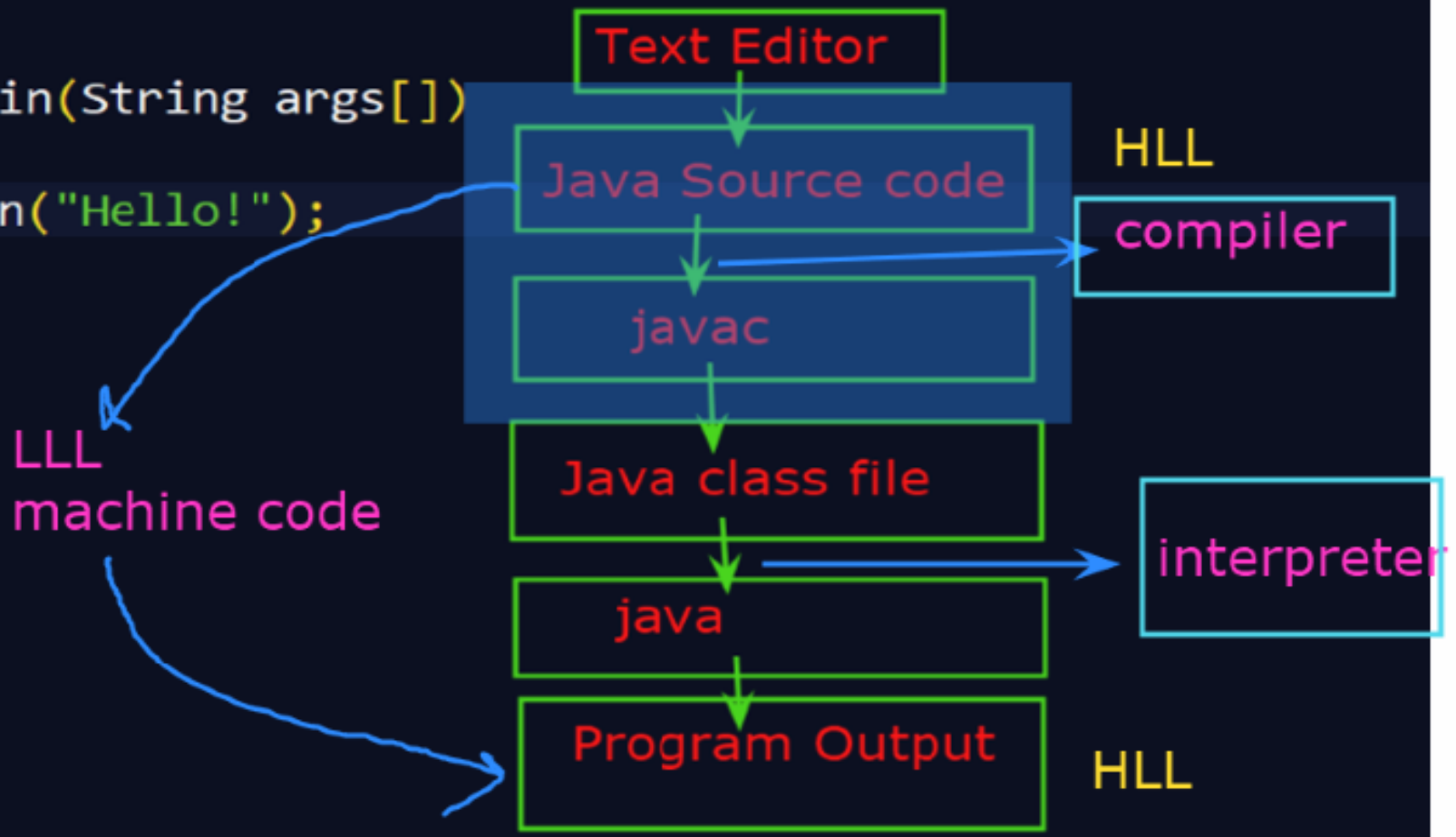
Concepts of Programming
Day 2: Mar 2023

Introduction to Java

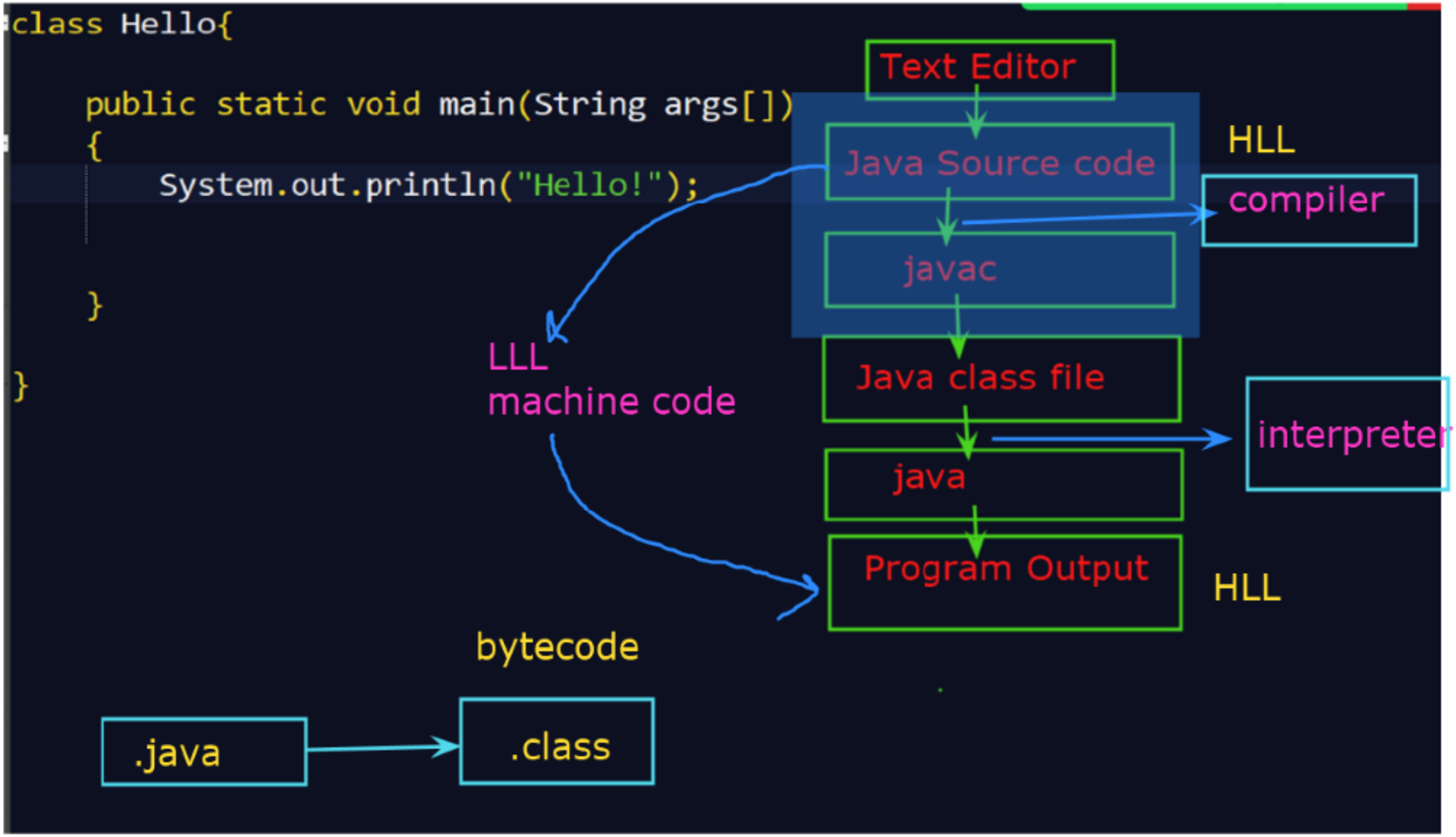
Kiran Waghmare

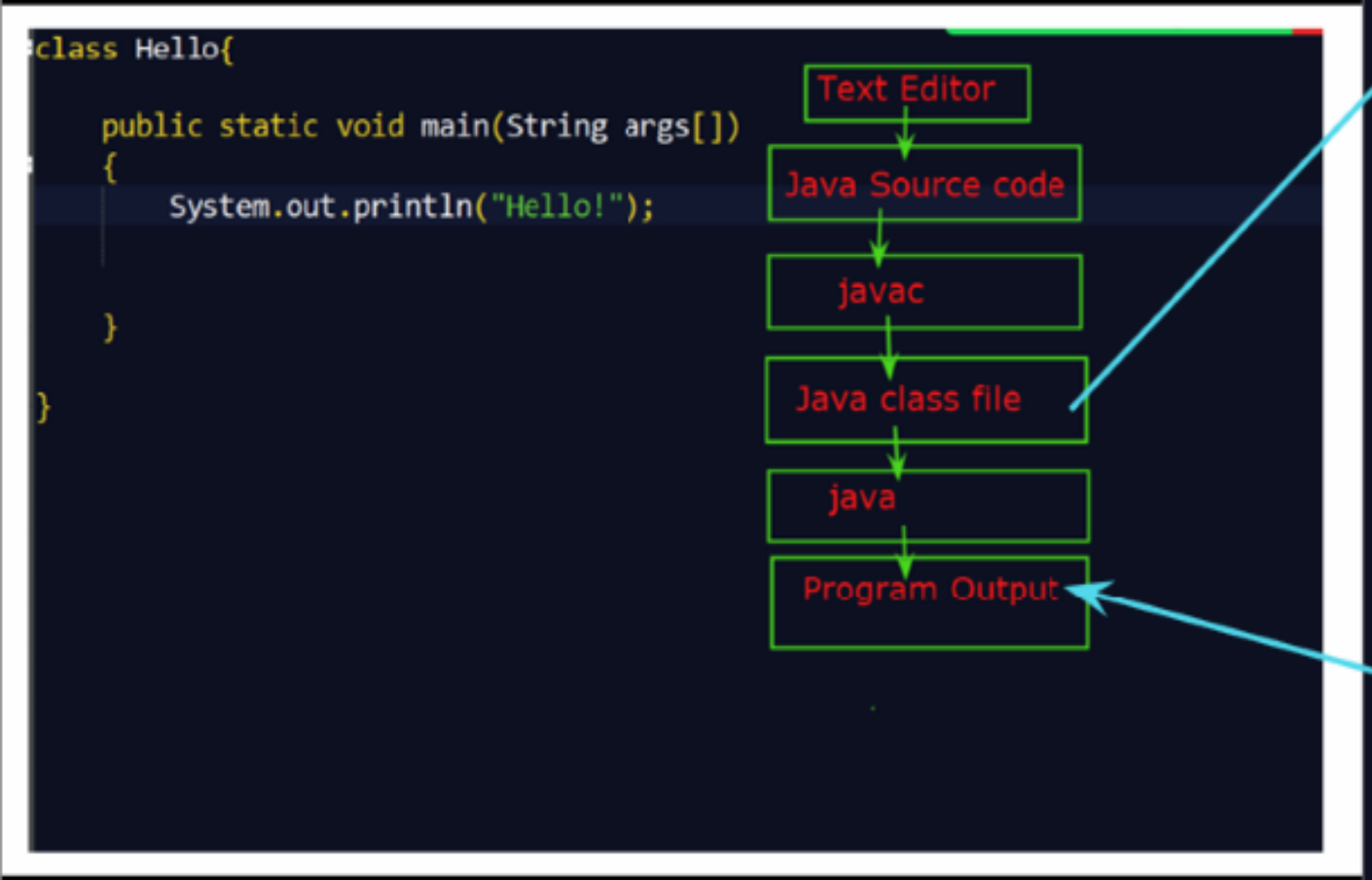
CDAC Mumbai

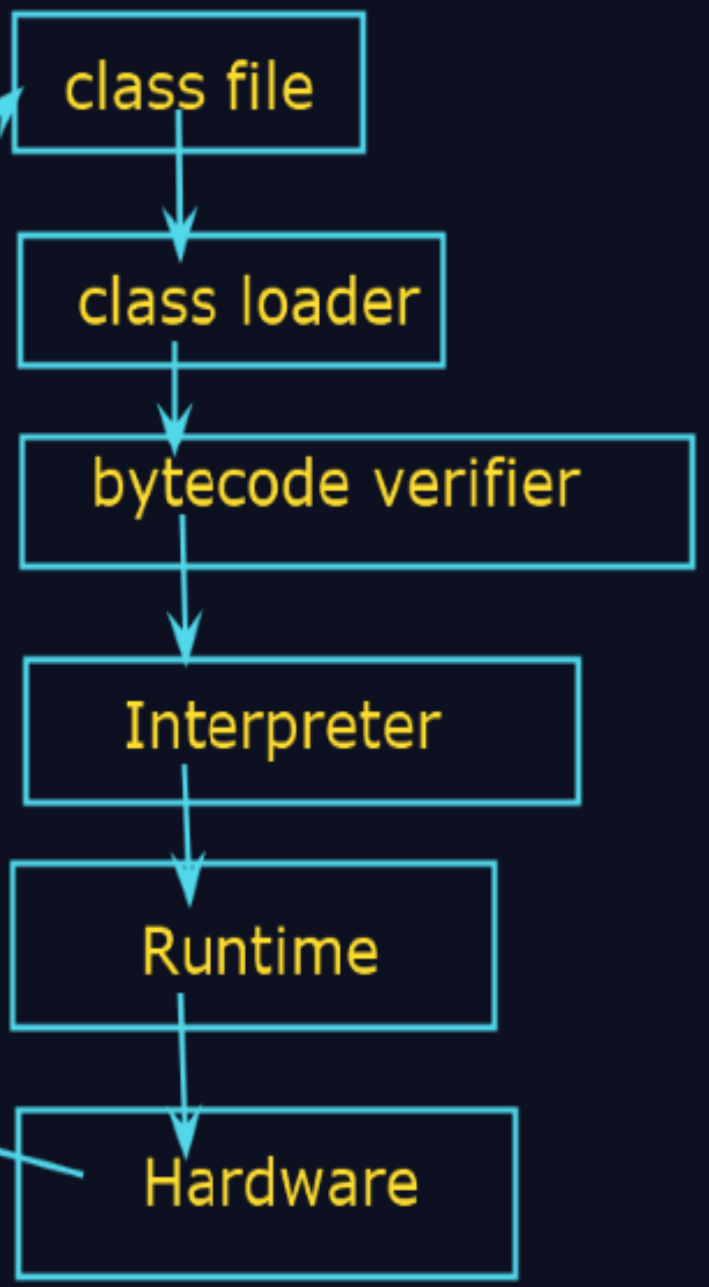
```java
class Hello{

    public static void main(String args[])
    {
        System.out.println("Hello!");


    }

}
```



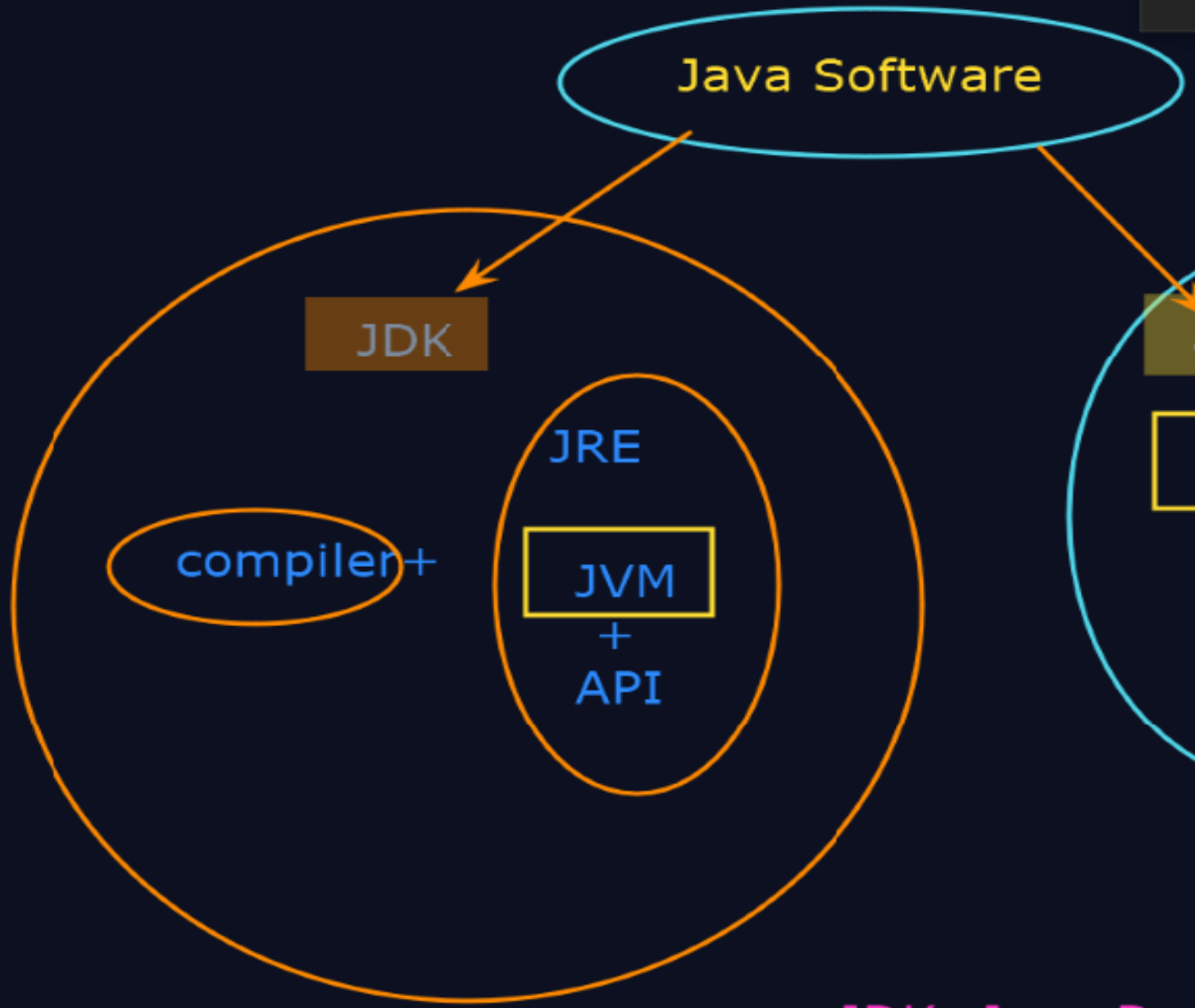
```java
class Hello{

    public static void main(String args[])
    {
        System.out.println("Hello!");


    }

}
```




class Hello{
public static void main(String args[])
{
System.out.println("Hello!");
}
}
Text Editor
Java Source code
javac
Java class file
java
Program Output


class file
class loader
bytecode verifier
Interpreter
Runtime
Hardware

Developer:D, C, E
User:E



JVM:

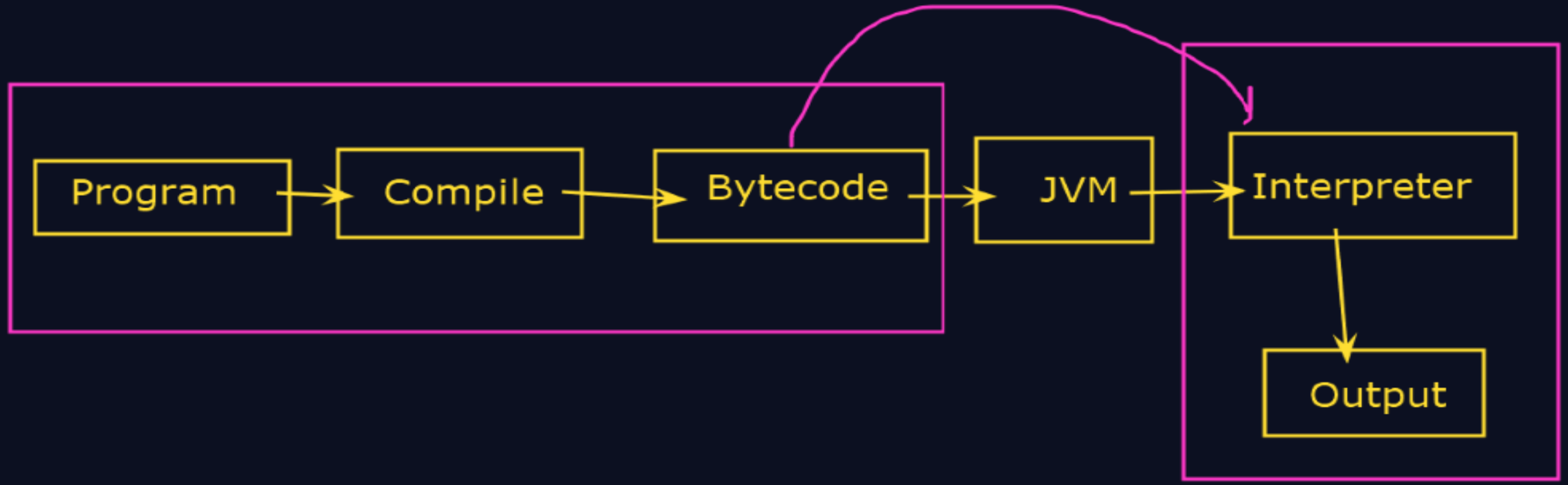
-abstract machine
-Runtime env.
-H/w & S/w
-Follws the step:
-Load code
-bytecode verifie
-Execute code
-Runtime env.

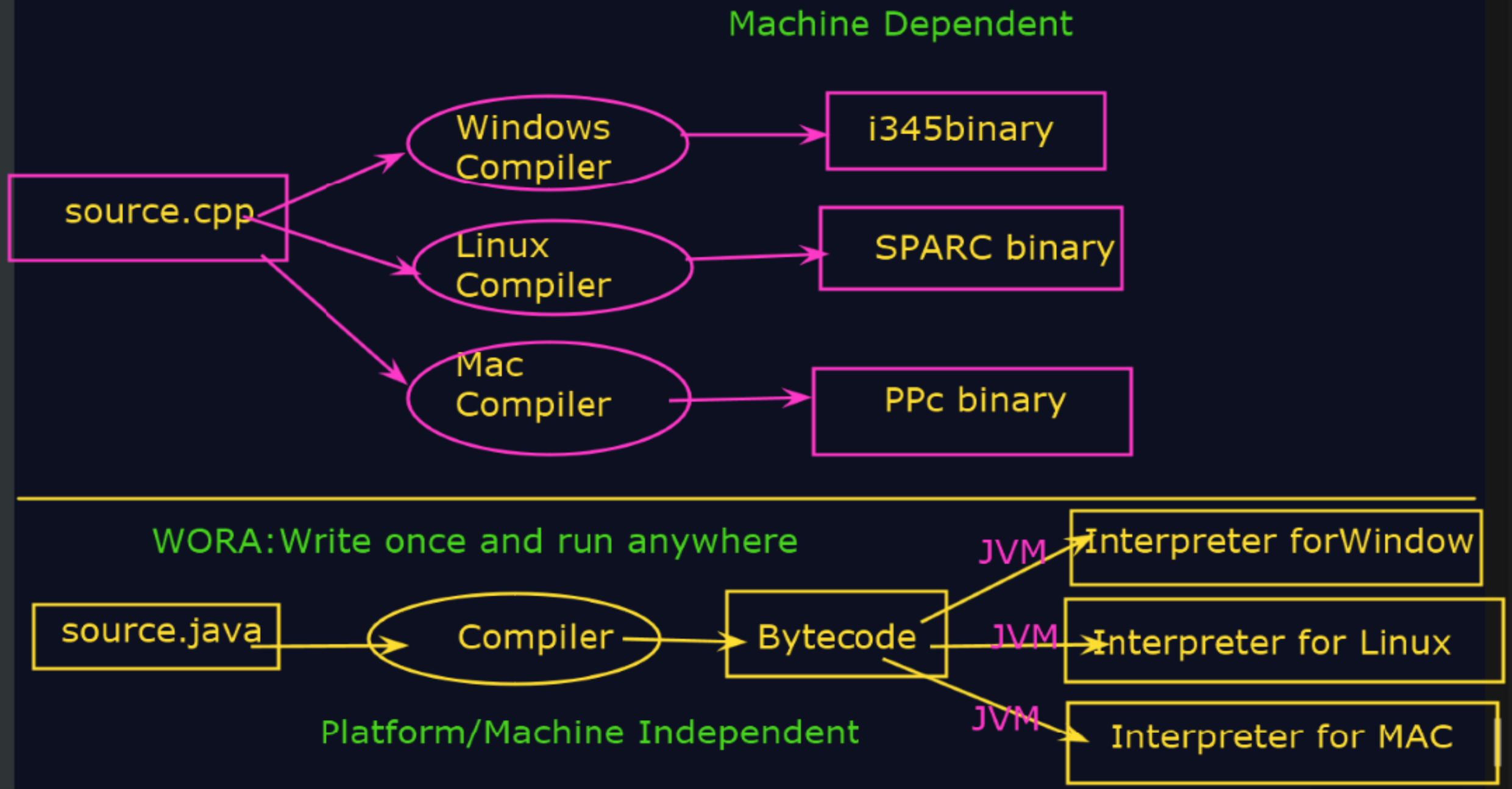
JDK: Java Development Kit
JRE: Java Runtime Environment
JVM: Java Virtual Machine

Day 6:Concepts of Programming

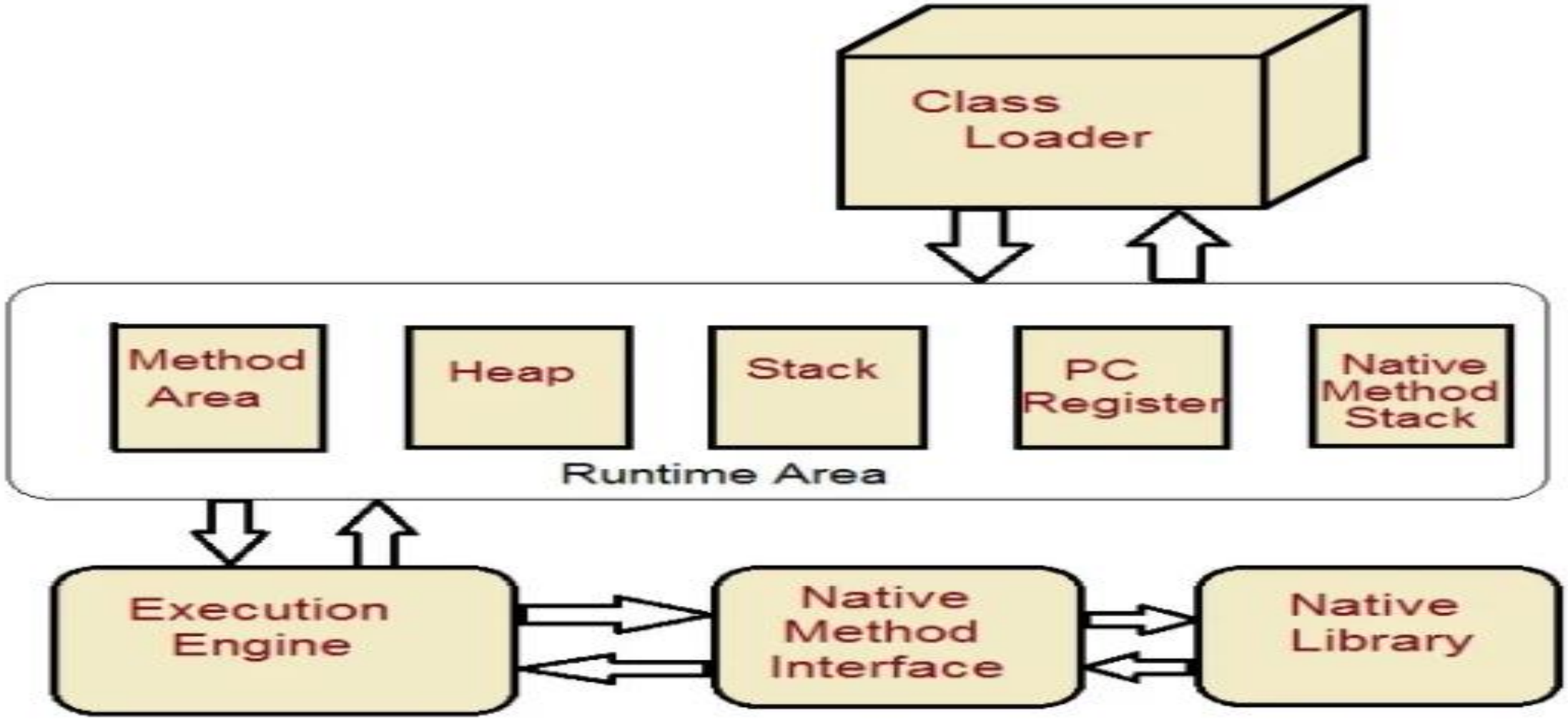
Topic:

- Introduction to Java Programming
- OOPs concept
- Compile and execution process




Machine Dependent
source.cpp
Windows Compiler
Linux Compiler
Mac Compiler
i345binary
SPARC binary
PPc binary
WORA:Write once and run anywhere
source.java
Compiler
Bytecode
JVM
JVM
JVM
Interpreter forWindow
Interpreter for Linux
Interpreter for MAC
Platform/Machine Independent

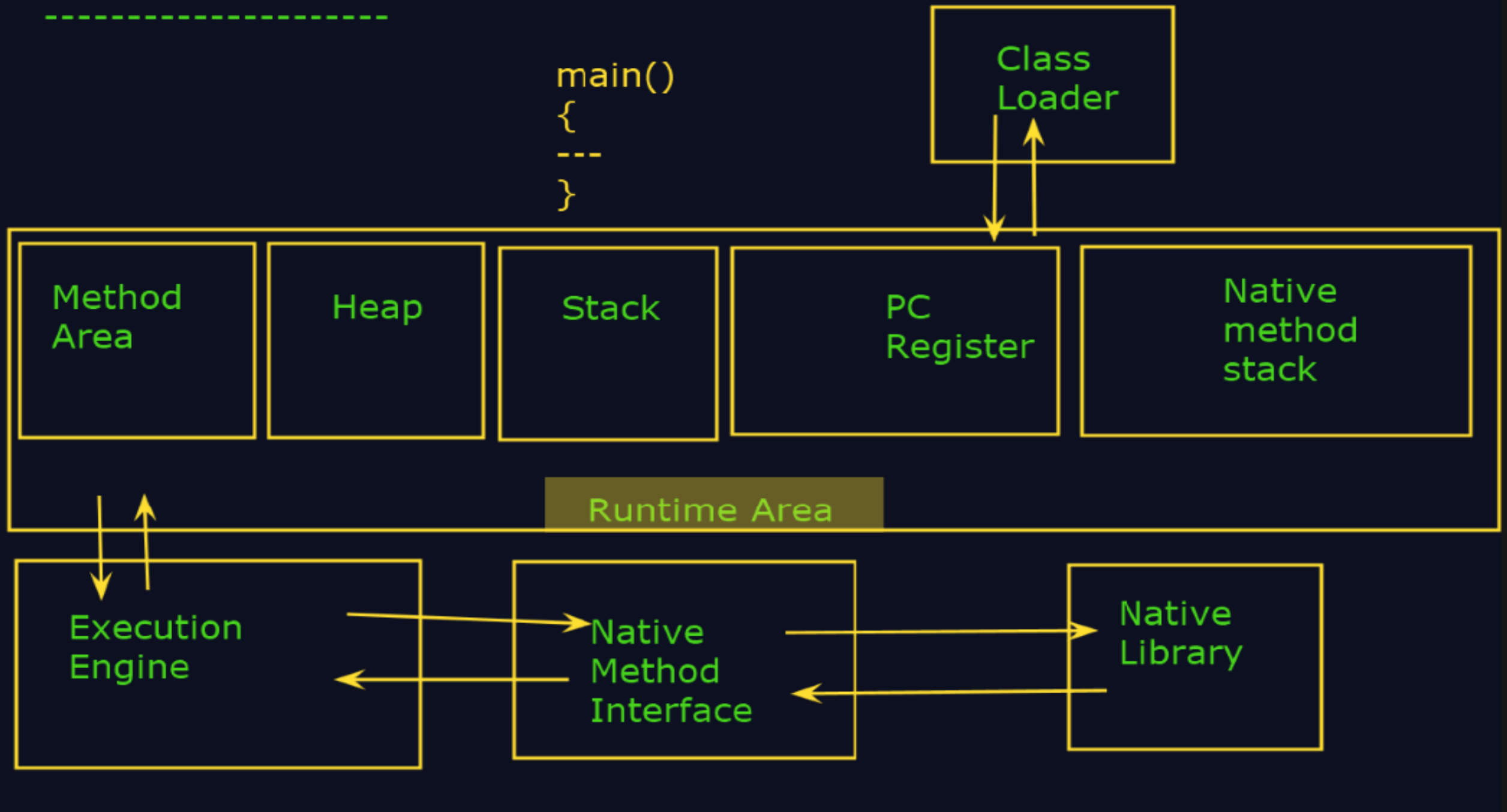
JVM Architecture



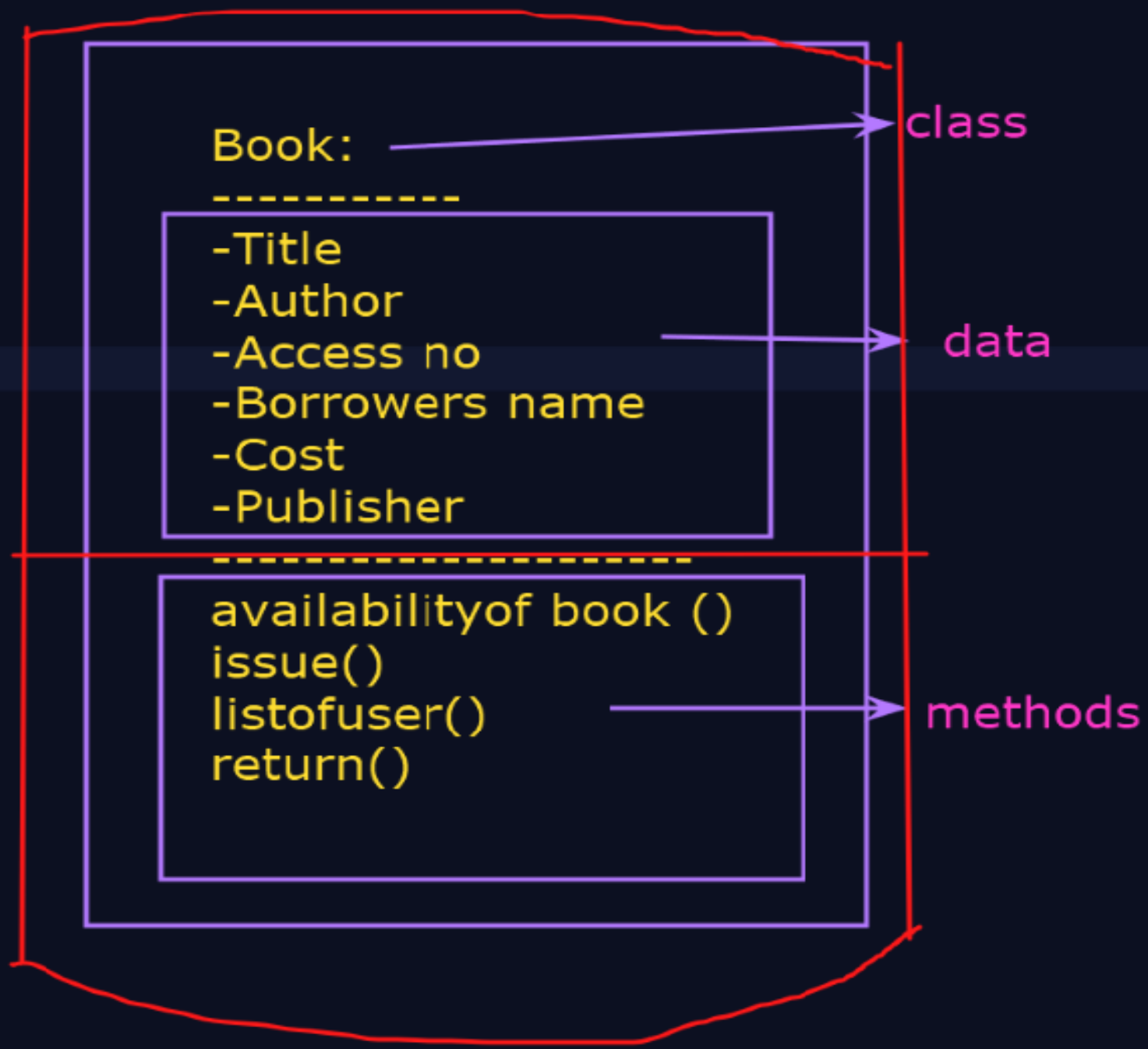
JVM Architecture:

JDK=Compiler + JRE (JVM + Lib/API)

```
main()
{
---
}
```



```
Features of Java:
-----------------

OOP pillars:
------------
1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism
```



Features of Java:

OOP pillars:
---
1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

Book

Text books

-category

Reference book

-facultylist

Reusabilty

Book:

-Title
-Author
-Access no
-Borrowers name
-Cost
-Publisher

availabilityof book ()
issue()
listofuser()
return()

class

data

methods

Features of Java:

OOP pillars:
---
1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

Access specifier:

-Private
-Public
-Protected
-default

pay=****

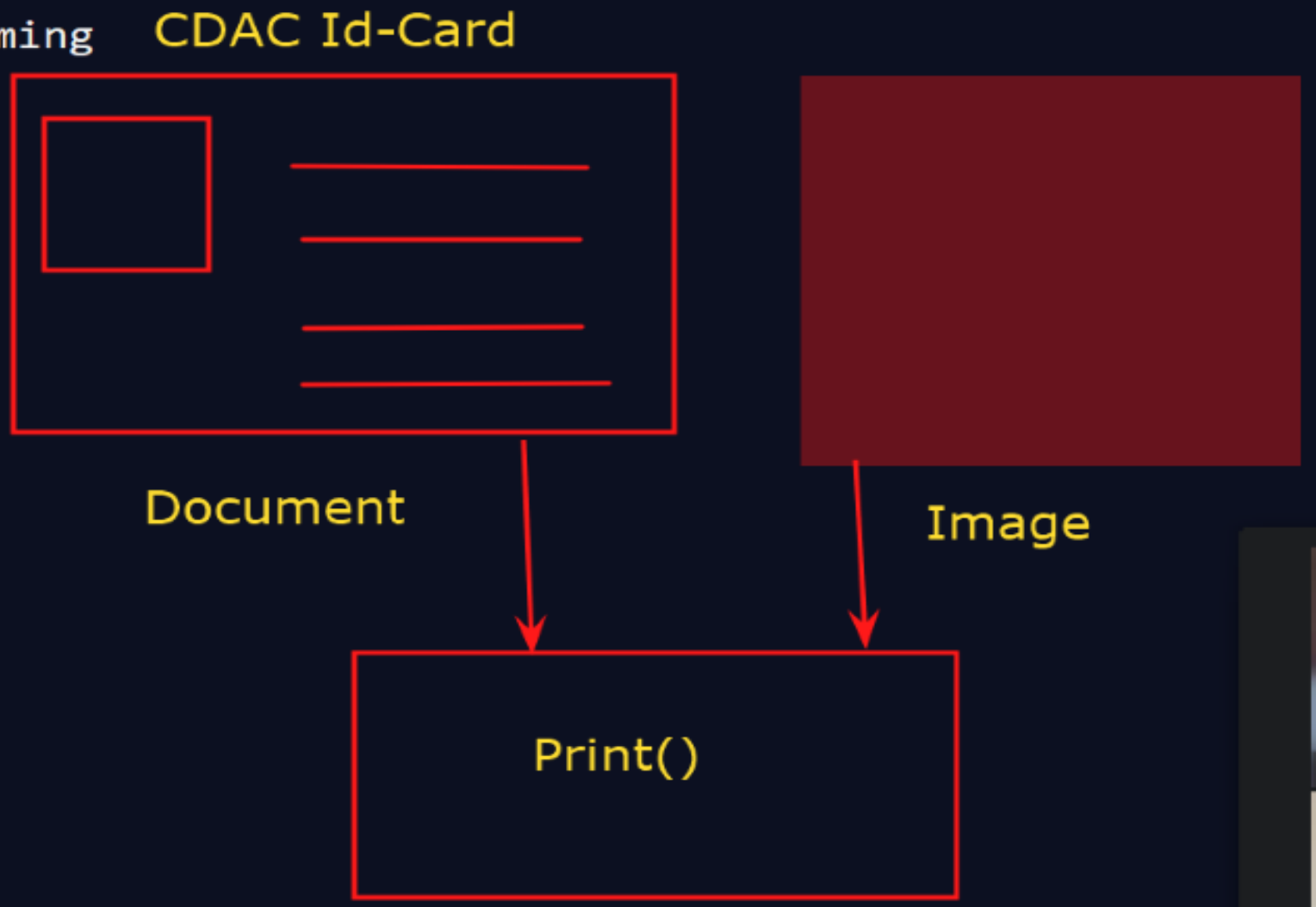
Information hiding
(Data hiding)

Topic:
    -Introduction to Java Programming
    -OOPs concept

Features of Java:

1.Simple
2.Object oriented
3.Platform independent
4.Robust
5.Secure
6.Portable
7.Interpreted
8.Multhreded
9.Architectural neutral
10.Distributed
11.Dynamic
12.High performance

OOP 4 pillars:
---
1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism



Topic:
    -Introduction to Java Programming
    -OOPs concept

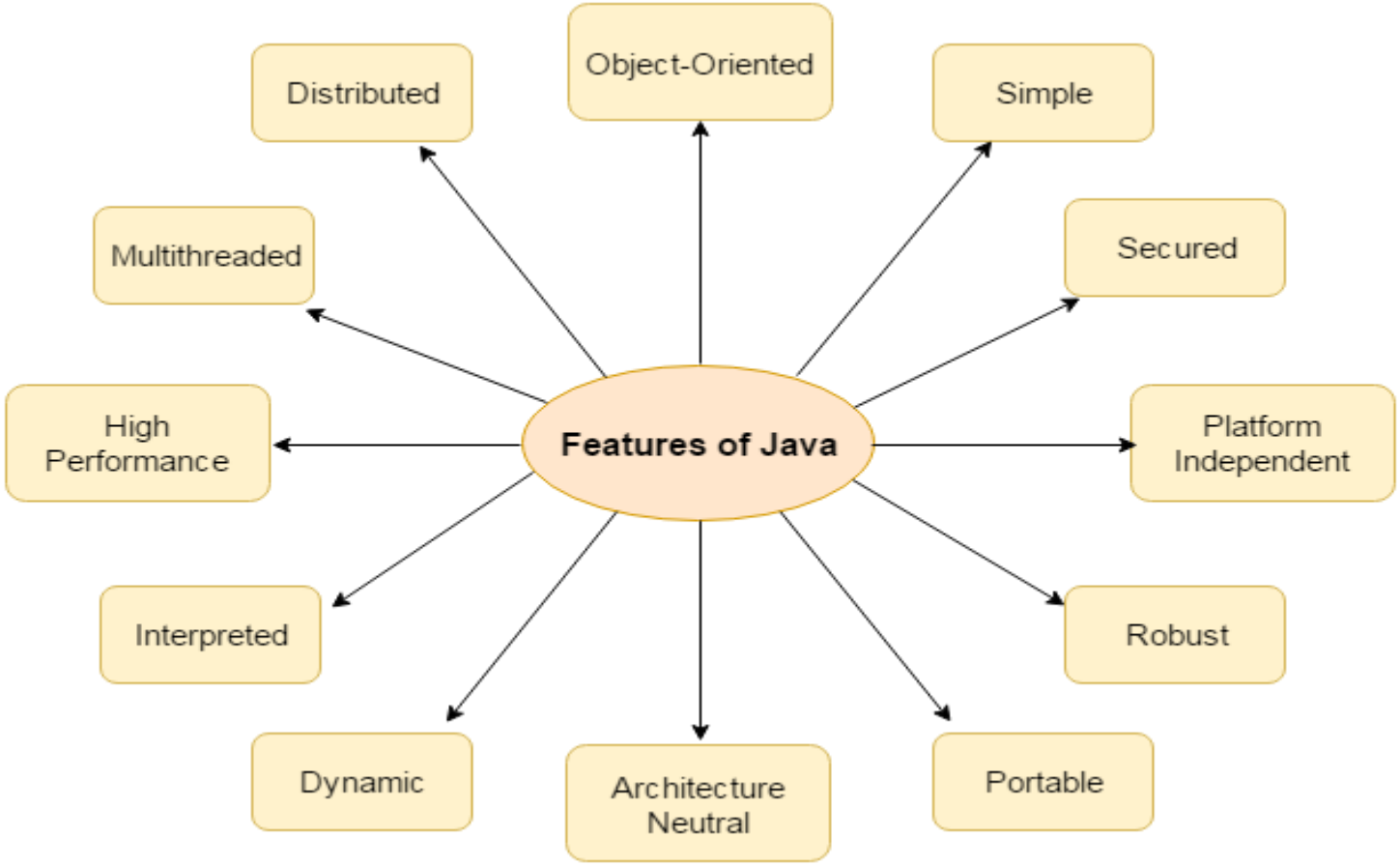
Features of Java:

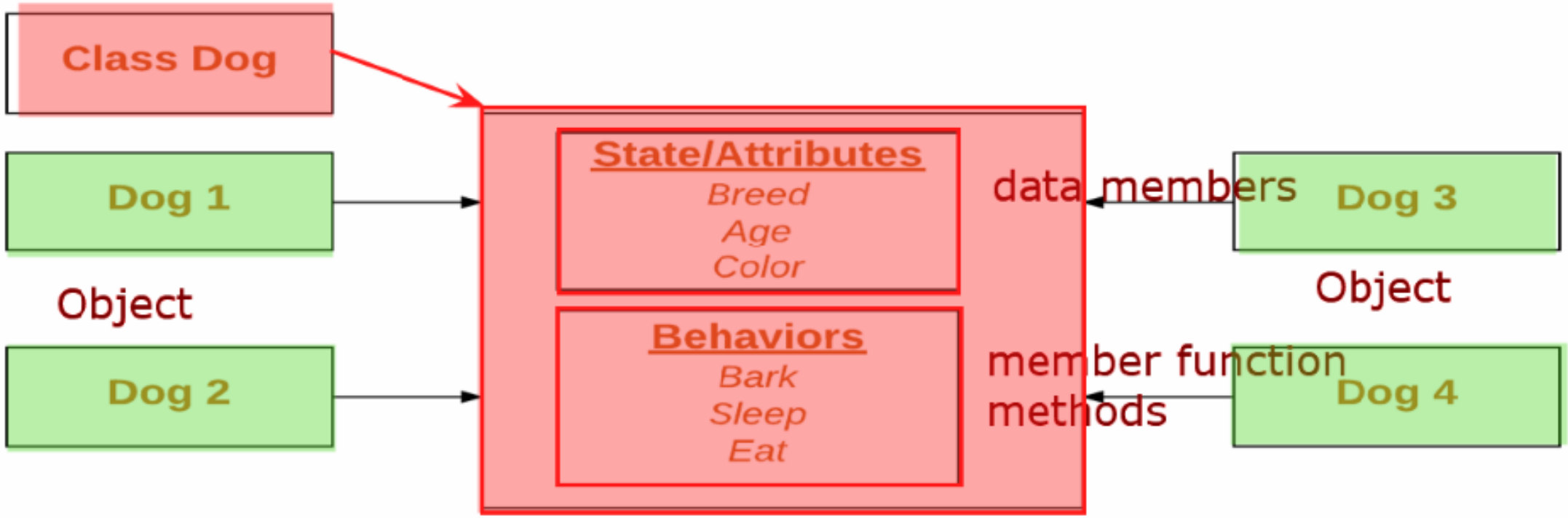
1.Simple
2.Object oriented
3.Platform independent
4.Robust
5.Secure
6.Portable
7.Interpreted
8.Multhreded
9.Architectural neutral
10.Distributed
11.Dynamic
12.High performance

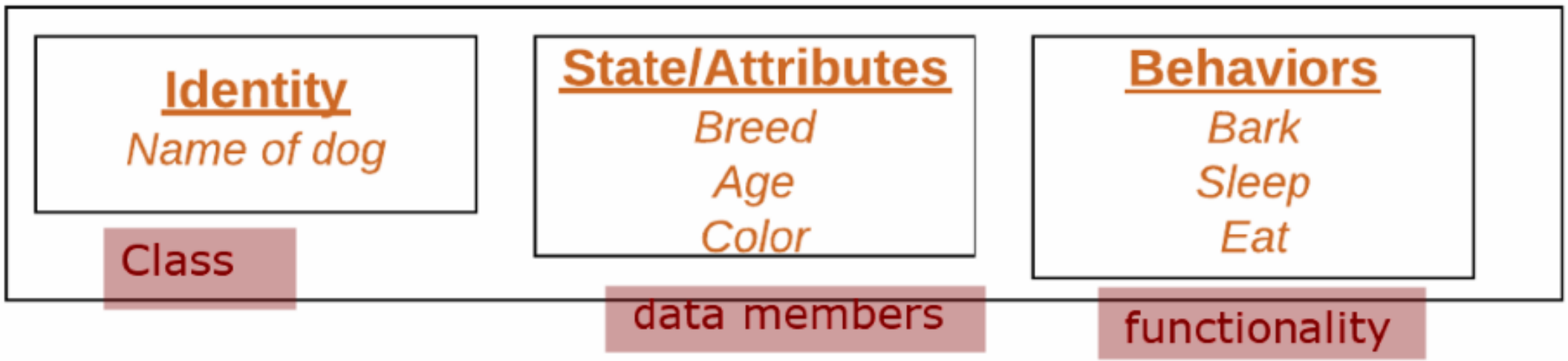
```
add(x,y);// 12+34

add(x,y);//"Kiran"+"Waghmare"

add(x,y)'//Image+document

add(x,y);//doc1+doc2;
```

OOP 4 pillars:
---
1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

Features of Java(12)




Class Dog
Dog 1
Dog 2
Object
State/Attributes
Breed
Age
Color
Behaviors
Bark
Sleep
Eat
data members
member function
methods
Dog 3
Dog 4
Object

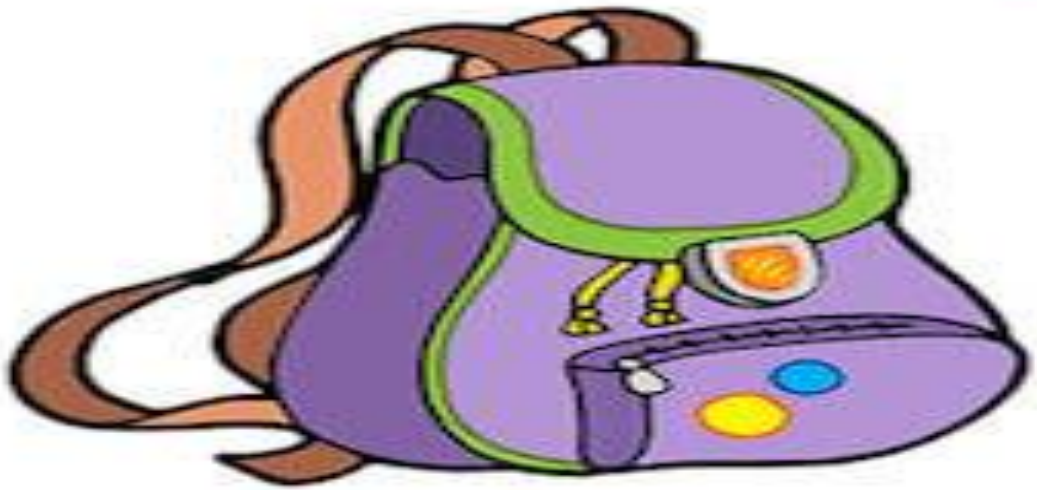

Identity
Name of dog
Class
State/Attributes
Breed
Age
Color
data members
Behaviors
Bark
Sleep
Eat
functionality

Objects: Real World Examples

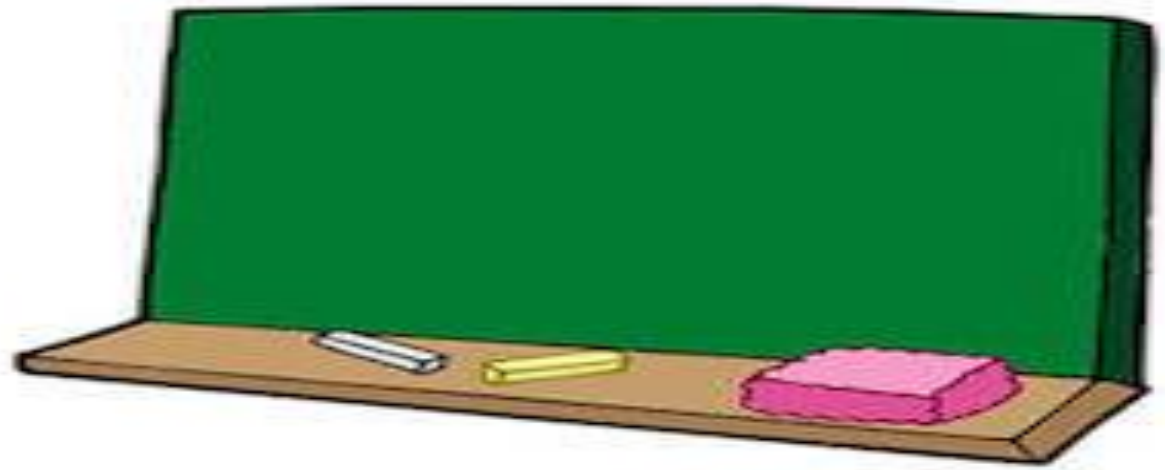
Pencil

Apple

Book



Bag



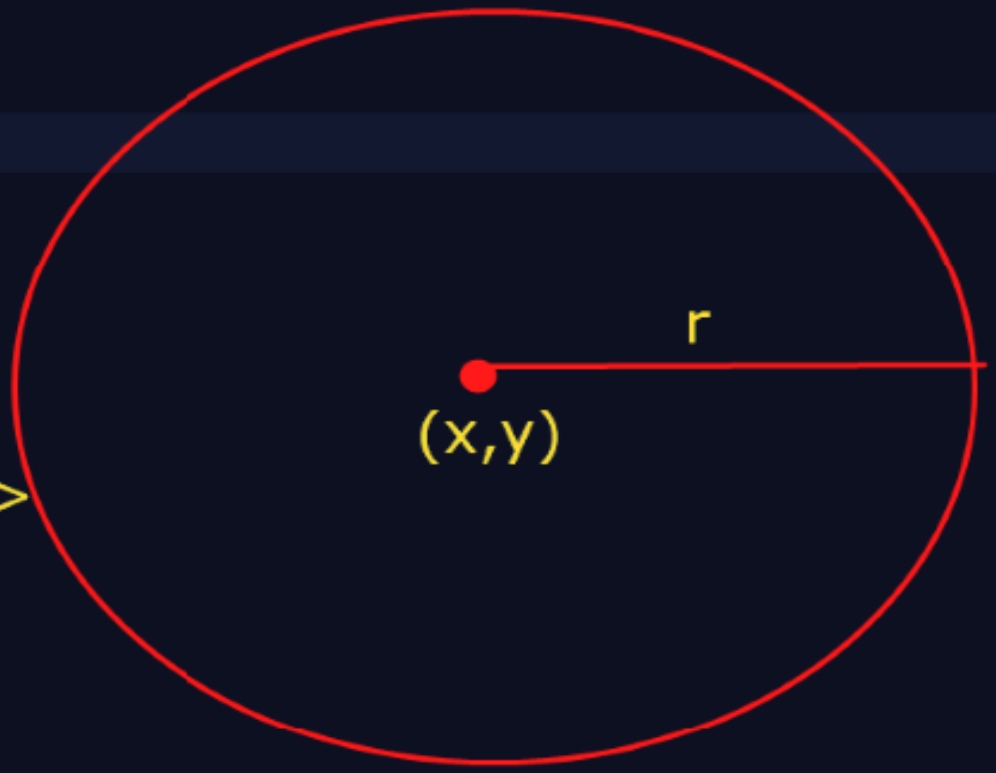
Board

Access specifier:

-Private
-Public
-Protected
-default

Class and Object:

Logical template for Area of circle===>



r=5;
r=10;

Area of circle=pi*r*r;

Input= r;
pi=3.14;

```
class Employee{

    //data members
    String name;
    int empId;
    int age;

    //methodname

    void empInfo(){

    }

}
```

Hoework:

Q1. Create a class Actor and add data menbers and methods.

Q2. Create a class Cake and add data menbers and methods.

Q3. Create a class Radio and add data menbers and methods.

Q4. Create a class Bank and add data menbers and methods.

```
syntax:
--------
class classname{

    //data members or variables

    //member function or methods

}
```

template

```
class Employee{

    //data members
    String name;
    int empId;
    int age;

    //methodname

    void empInfo(){

    }

}
```

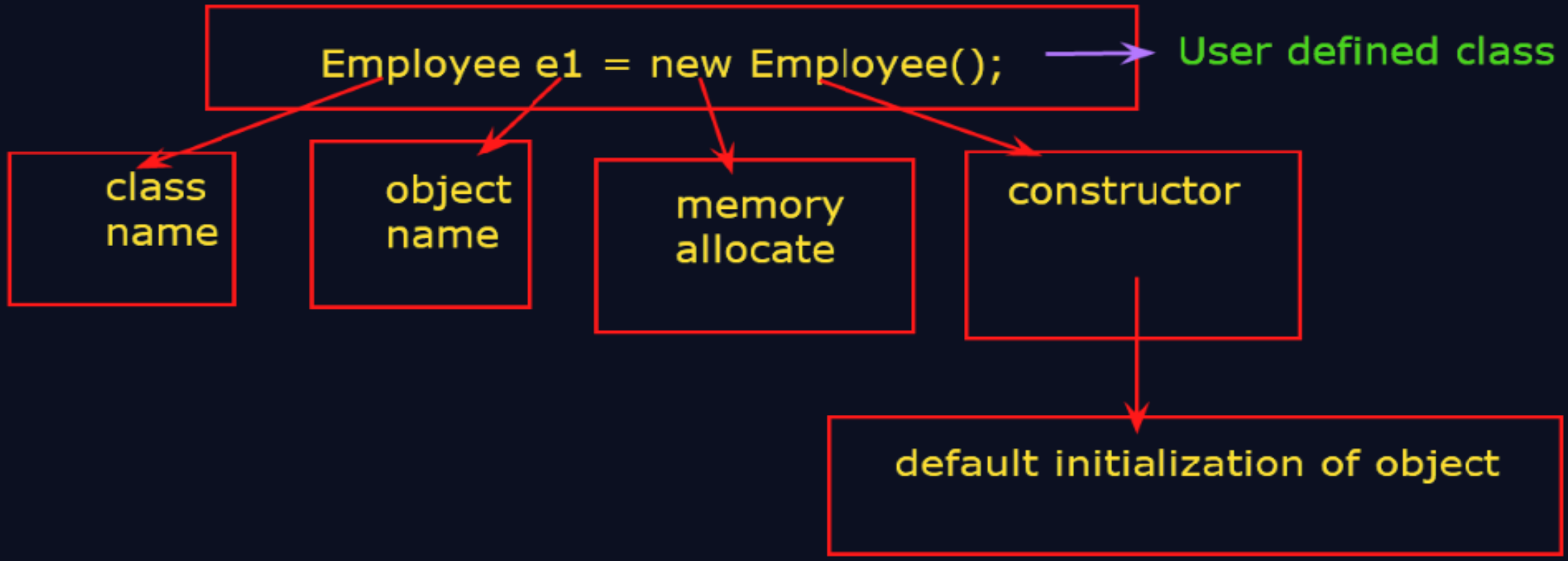
instance

e1
e2
e3
e4
e5

Object :real life entities

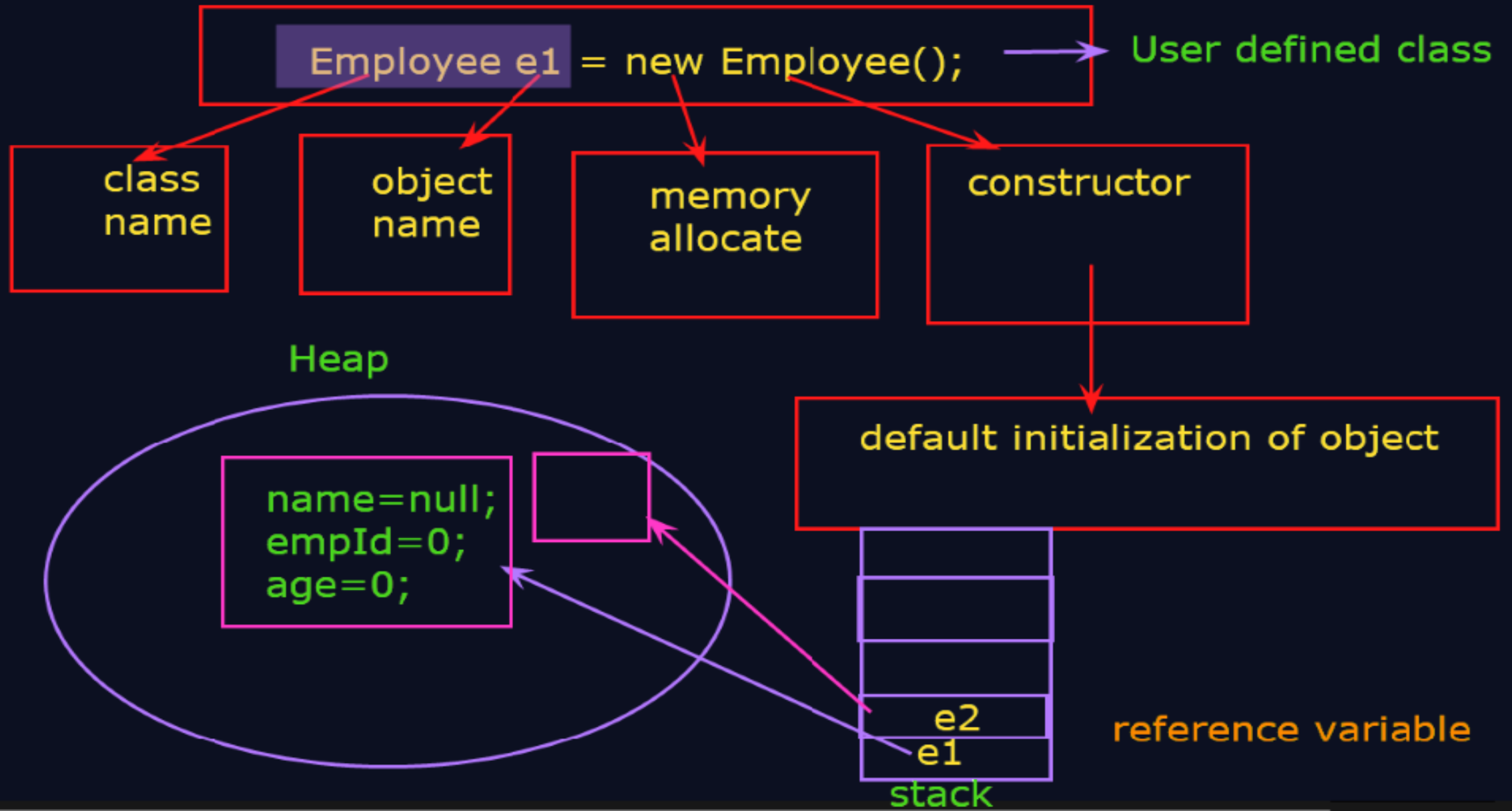
Example:

```
Employee e1 = new Employee();
```

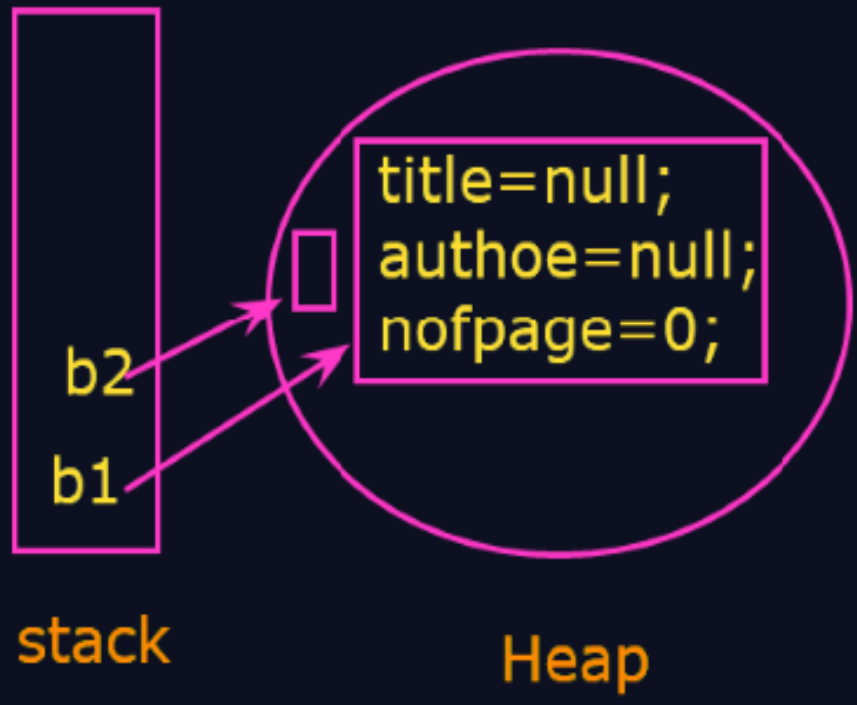


Example:

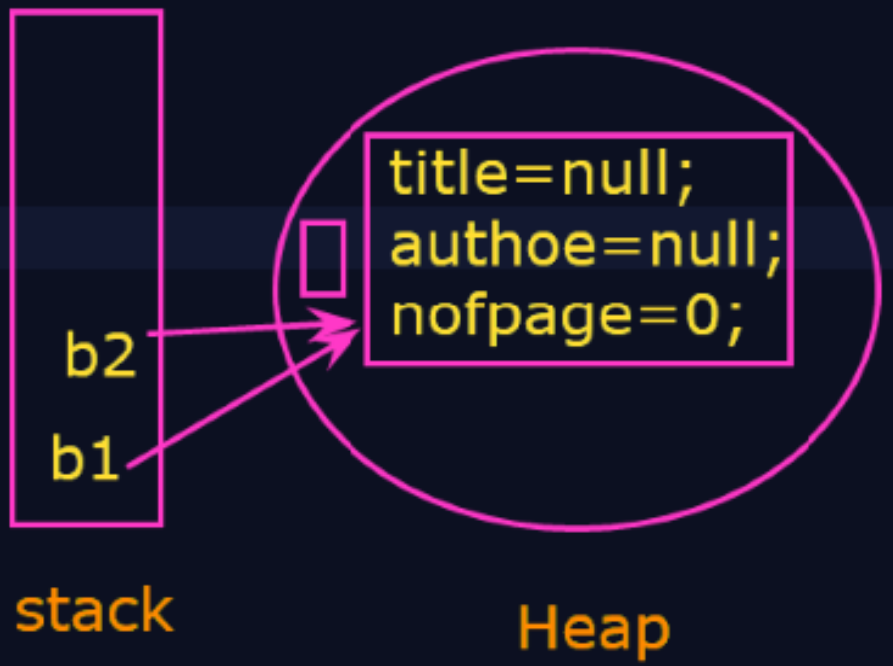
```
Employee e1 = new Employee();
```



```java
class Book{
    //instance variable
    String title = "Let us C";
    String author="Y Kanetkar";
    int nofpages= 300;

    public static void main(String args[])
    {
        //Local variable
        //String title = "Let us C";
        System.out.println("Object creation !!!");
        System.out.println(title);

        Book b1 = new Book();
        Book b2 = new Book();


    }

}
```



```
class Book1{
    //instance variable
    String title = "Let us C";
    String author="Y Kanetkar";
    int nofpages= 300;

    public static void main(String args[])
    {
        //Local variable
        //String title = "Let us C";
        System.out.println("Object creation !!!");

        Book b1 = new Book();
        System.out.println(b1.title);
        System.out.println(b1.author);
        System.out.println(b1.nofpages);

        Book b2 = new Book();
        System.out.println(b2.title);
        System.out.println(b2.author);
        System.out.println(b2.nofpages);

    }
```



```
class Test1{
    int i=10;

    void display()
    {

        System.out.println("Function prints the value of i="+i);
    }

    public static void main(String args[])
    {

        Test1 t1 = new Test1();
        System.out.println(t1.i);//10
        t1.display();//Method call
        System.out.println(t1.i);//10
        t1.display();//Method call
        t1.display();//Method call

        Test1 t2 = new Test1();
        t2.i=20;
        t1.display();//Method call

    }
}
```