Employee -> manager,salesman

Upcasting and Downcasting

```
Employee e1 = new Manager(); // upcasting
e1.members??  // members from the super class inherited into the subclass
e1.subclass members ??  // NO -> Object Slicing

if(e1 instanceof Manager) // To prevent the ClassCastException
Manager m1 = (Manager)e1; // Downcasting
m1. subclass members
```

When downcasting fails java throws an exception ClassCastException

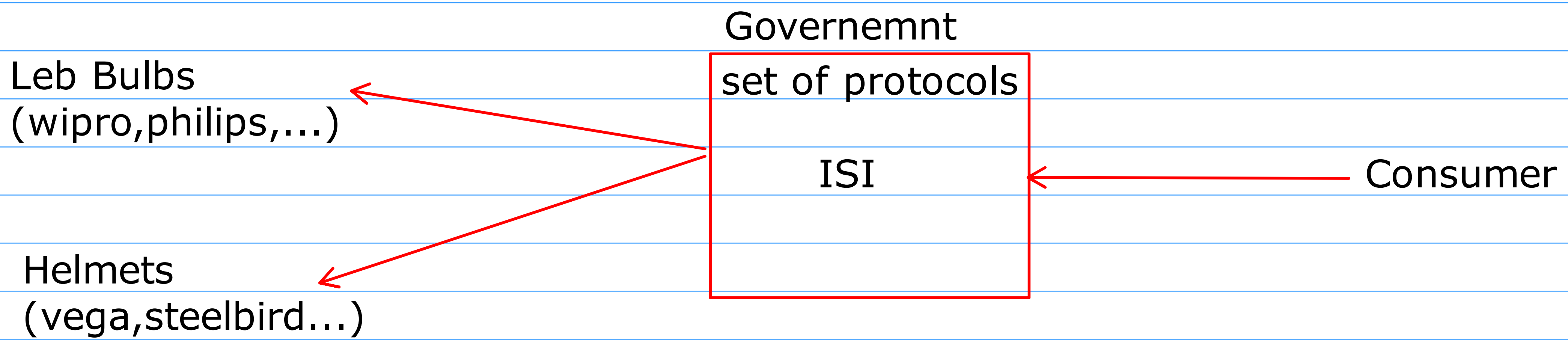
Why to do Method Overrding

1. Super class method implementation is partial complete (accept(), display())
2. Super class method implementation is 100% incomplete (calculateTotalSalary())
3. The implementation in the sub class we want totally differnent from the super class method (toString(), equals())

Hirerachy -> Reuse -> has-a, is-a

Interface - java 7
- It is set of protocols/ specification
- It provides a common method design in all the classes

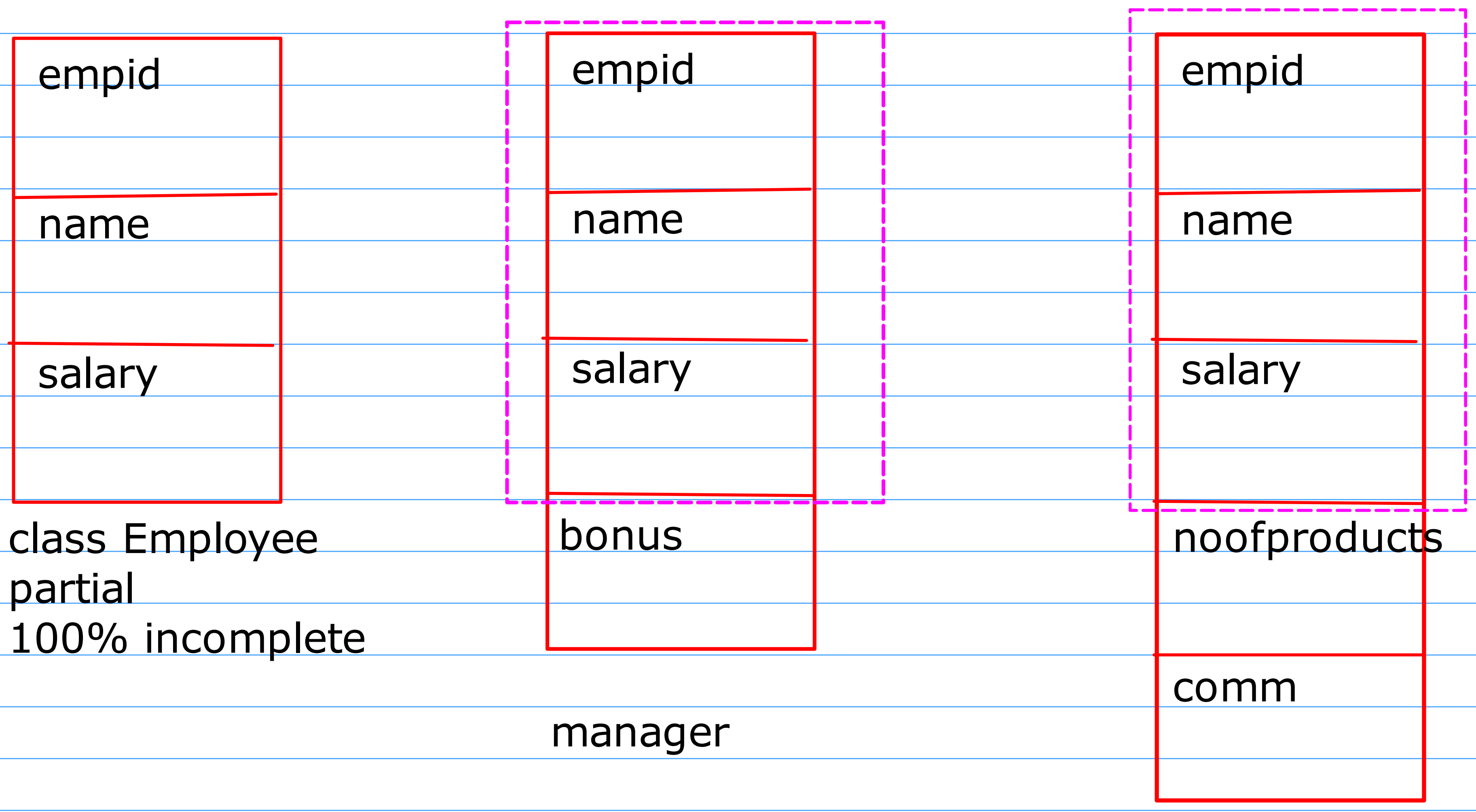
Suppliers



USA

Fragile Base Class Problem

common implementation

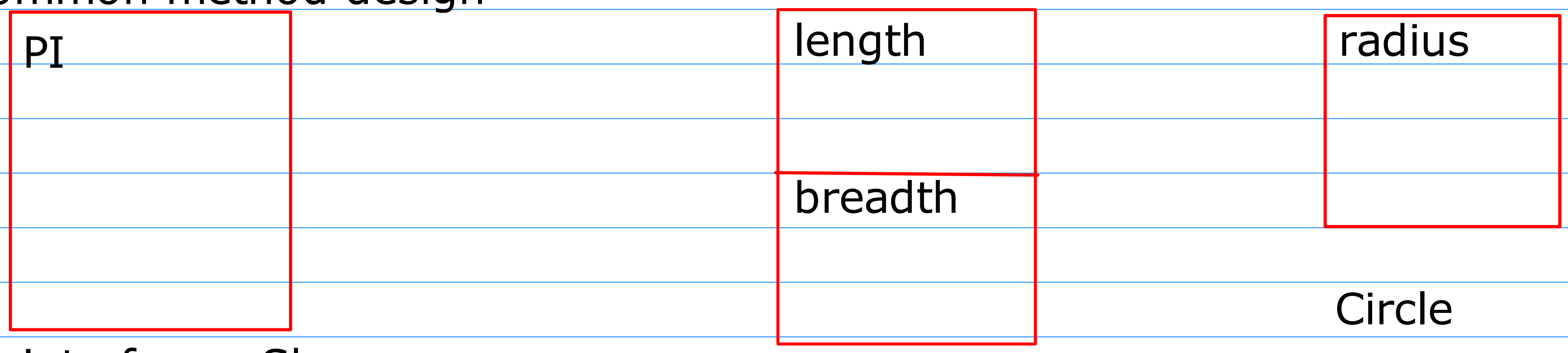


class Employee
partial
100% incomplete

manager

salesman

no common implementation
common method design



Circle

interface - Shape
100% incomple method

Rectangle

accept()
calculate()

accept()
calculate()

Method Area

```
interface Acceptable{
void accept();
void display();
}
```

```
class Date{
int day
int month
int year

}
```

```
class Product{
int pid;
String name;
double price;
Date manfacturing_date;
}
```

```
class Transport{
Product p1;
Product p2;
Product p3;
Date deliverDate;
}
```

java 7 interface
java 8 interface -> their are lot of changes... we will study during functional programming

# Types of inheritance between class and interface

```
//              class C3 extends C1 // OK
//              class C3 extends C1,C2 //NOT OK
                // Multiple class inheritance is not allowed in java

//              class C3 implements I1 // OK
//              class C3 implements I1,I2 // OK
                // Multiple interface inheritance is allowed in java

//              class C3 extends I1 // NOT OK
//              class C3 implements C1 // NOT OK

//              interface I3 extends I1 // OK
//              interface I3 extends I1, I2  // OK

//              interface I3 implements I1 // NOT OK
//              interface I3 implements C1 // NOT OK
//              interface I3 extends C1 // NOT OK
```

# Marker Interface

- An empty interface is called as a marker interface
- Marker interface also called as tagging interface are used to provide the extra information (metadata) or the permission for the JVM
- eg - Cloneable (clone() of object class after the exception) , Seralizable (File io)

JDK

- tools + docs + JRE(rt.jar + JVM)

runtime.jar (core libraries in the form of .class files)

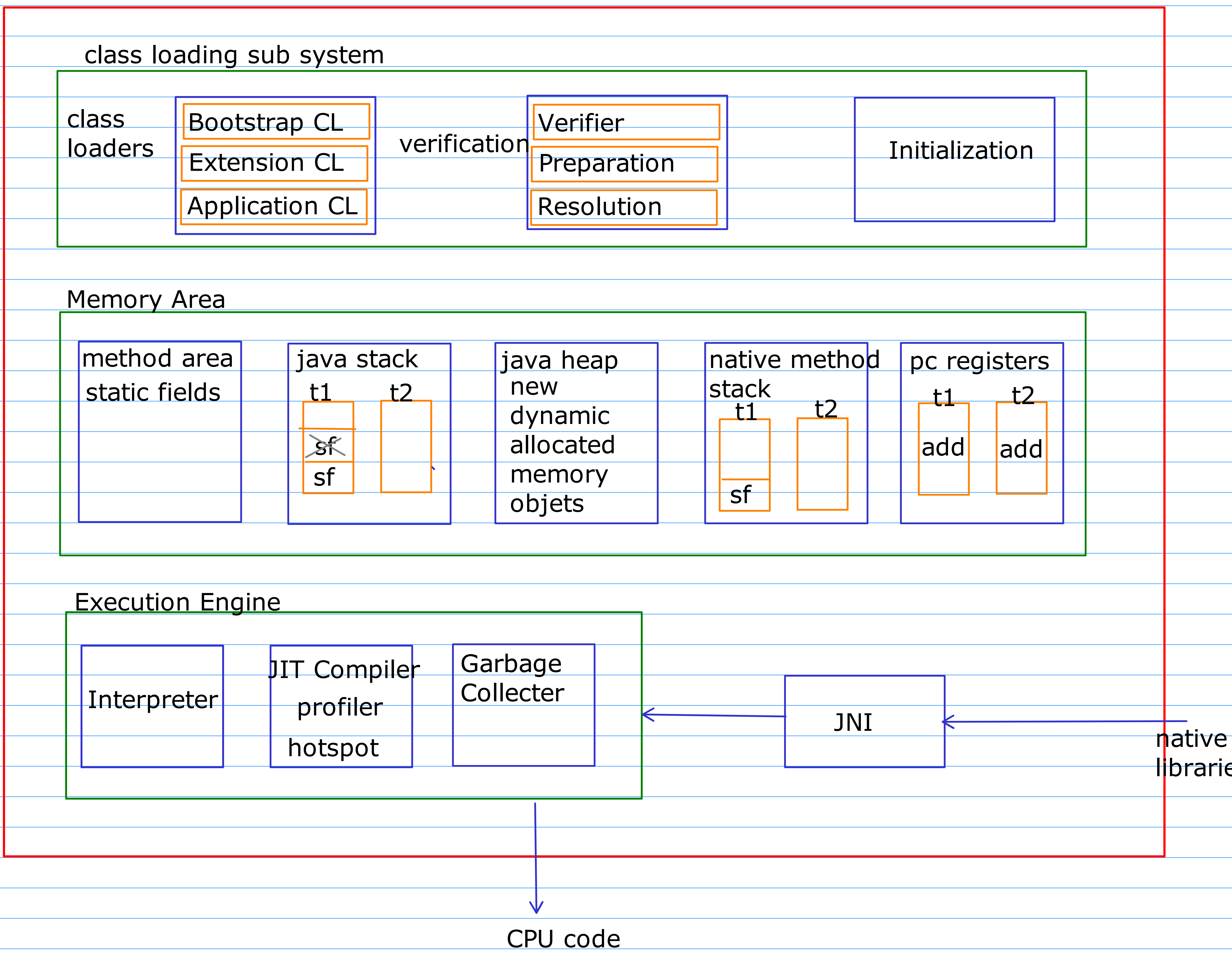
jre/lib/ext

.class -> javac   bytecode

new file -> .class

CAFEBABE

JVM -> Java Virtual Machine Architecture



accept() ->