Upcasting and Downcasting
- Super class reference can point at only the members of super class inherited into the sub class.
- This is called as object slicing
- The methods that get resolved by looking at reference are said to be having effect of early binding
- The methods that gets resolved by looking at the object that is created is said to be having effect of late binding
- In java by default late binding is done, i.e the methods to call gets resolved by looking at the object that is created.

- We cannot point at the members of sub class using super class reference.
- To access the members of sub class wh have to converst the reference of super class inti sub class reference
- this is called as downcasting.
- at the time of downcasting explicit type casting is mandatory
- If downcassting fails then jvm throws an exception ClassCastException

Object class
- Super class of all the classes in java
    - toString
    - equals

```
Employee e1 = new Employee();
//Employee e2 = e1;
Employee e2 = new Employee();
e1==e2 -> // true/ false
boolean equals(Object obj)
{
if(obj == null)
return false;
if(this==obj)
return true;
if(obj instance of Employee)
{
Employee e = (Employee)obj;
return (e.id ==this.id && e.sal==this.sal)
}
 return false;
}
```

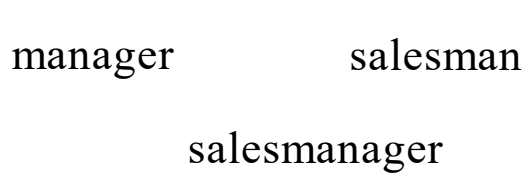
Abstract
1. Method
    - If the implementation of method is 100% incomplete, then such methods should be declared as abstract.
2. class
    - Abstract methods can be declared only inside abstract class
    - for an abstract class we cannot create the object however we can create its reference
    - An abstract class can have static as well as non static fields
    - we can also declare ctor inside abstract class
    - Abstract classes are used to group related types together.

Fraglie Base Class Problem
    - To avoid this problem use interface
Interface
    - It is set of rules/protocols/specifications provided for the classes
    - The methods we declared inside the interface are by default public and abstract

```
interface Acceptable{
void accept(Scanner sc);
}
```

```
class Employee implements Acceptable{

}
```



Shape

```
static const  PI;
virtual void accept()=0;
virtual void calculateArea()=0;
```

Circle

radius

Rectangle

length, breadth

interface Shape

```
double PI = 3.14;
void accept(Scanner sc);
void calculateArea();
```

```
class Circle implements Shape{
double radius;
}
```

```
class Rectangle  implements Shape{
int length;
int breadth;
}
```

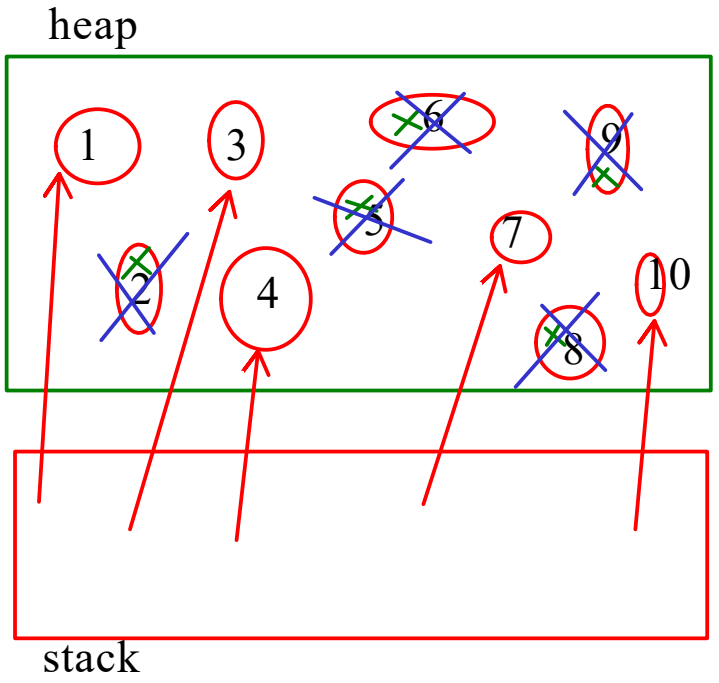
```
// Marker interfaces
interface emptyInterface{

}
```

Marker Interface
- An interface that does not have any method inside it is called as marler interface
- It is used to provide extra information/ meta data to the JVM regarding the class that implements this interface
- These are also called as Tagging interface

Garbage Collector
- Two threads starts
1. main thread
2. gc thread

Mark and Compact Algorithm



how do the Garabage collector identify the objects whose references are missing?

- case 1

```
Time t1 = new Time(); // GC
t1 = null;
```

- case 2

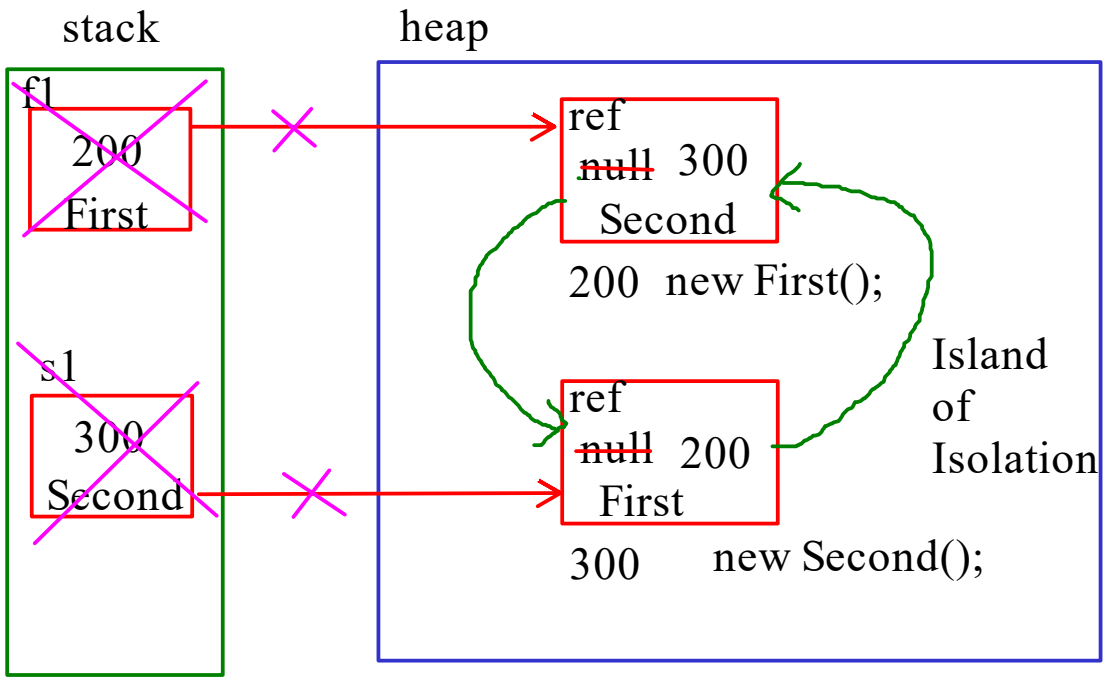
```
Time t1 = new Time(); // GC
Time t2 = new Time();
t1 = t2;
```

- case 3

```
Time t1 = new Time(); // GC
t1 = new Time();
```

- case 4

Island of Isolation




```
class First{
Second ref;
}
```

```
class Second{
First ref;
}
```

```
First f1 = new First();
Second s1 = new Second();
```

```
f1.ref = s1;
s1.ref = f1;
```

Stack

Heap

t1

200

null

Time

hrs | min

200

new Tim();

t1

200

300

Time

hrs | min

200

new Tim();

t1 = t2;

t2

300

Time

hrs | min

300

new Tim();

t1

200

300

Time

hrs | min

200

new Tim();

Time t1 = new Time();

t1 = new Time();

hrs | min

300

new Tim();

Heap

Young Generation

Old Generation

1

8

2

3

9

4

5

6

7

1. Minor GC
2. Major GC

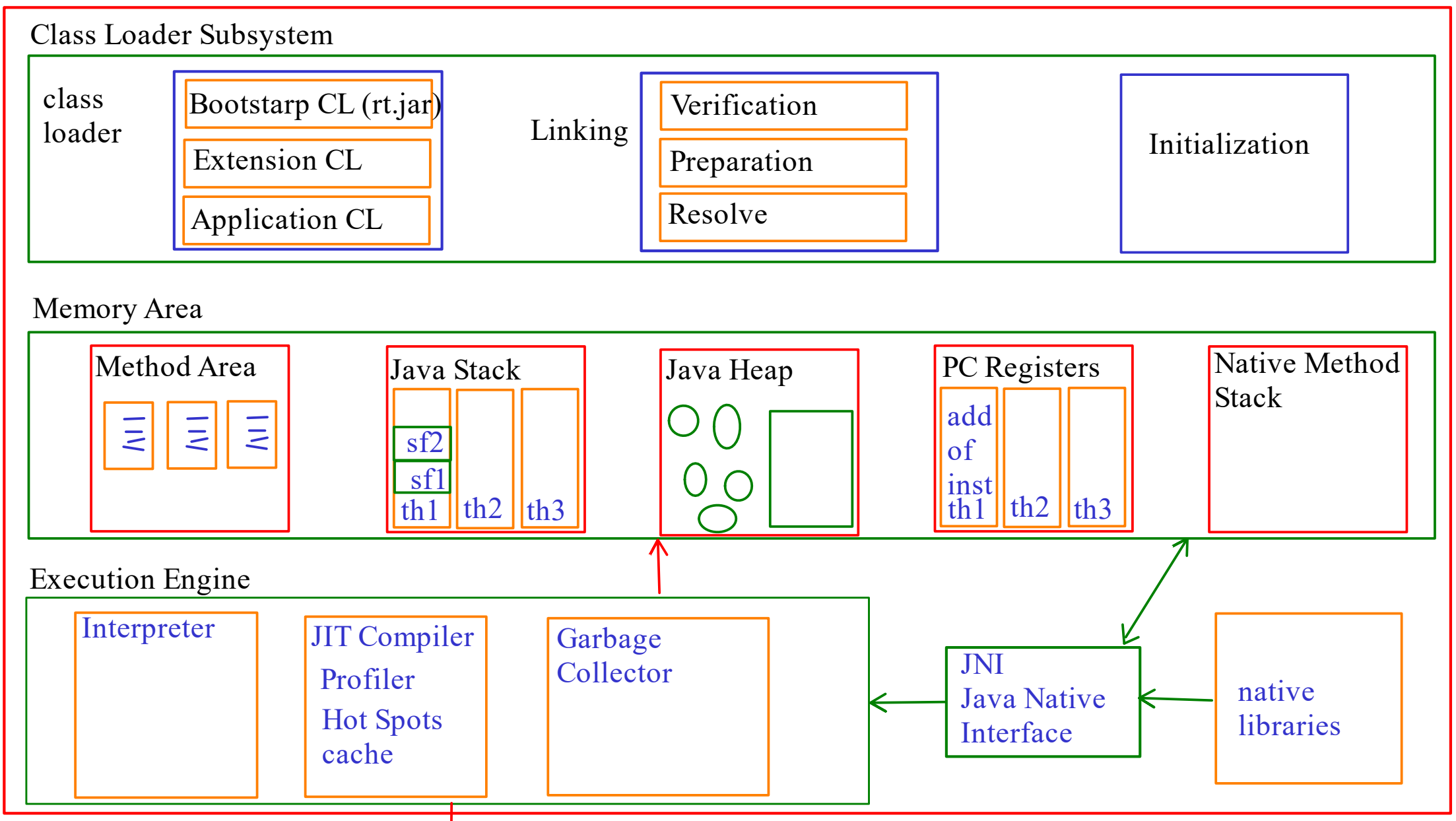
```
System.gc();
Runtime.getRuntime().gc();
```

```
class Time{

@override
protected void finalize(){
sysout("Inside time finalize");
}
}
```

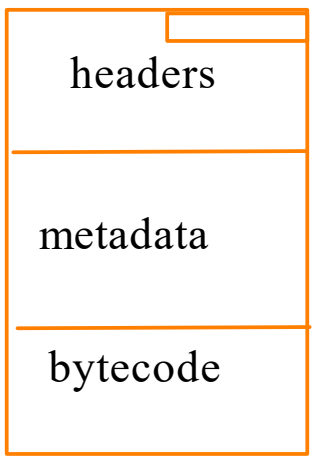
```
Time t1 = new Time();
t1 = null;
System.gc();
```

JVM -> Java Virtual Machine



CPU code

.class



java Program