Revison
Hirerachy
- has-a (Association) (Composition, Aggegration)
- is-a (Inheritance)

eg- Employee has-a Date

```
//Dependancy class
class Date{

}

//Dependent
class Employee{
Date doj; // Association
}
```

Employee is-a Person

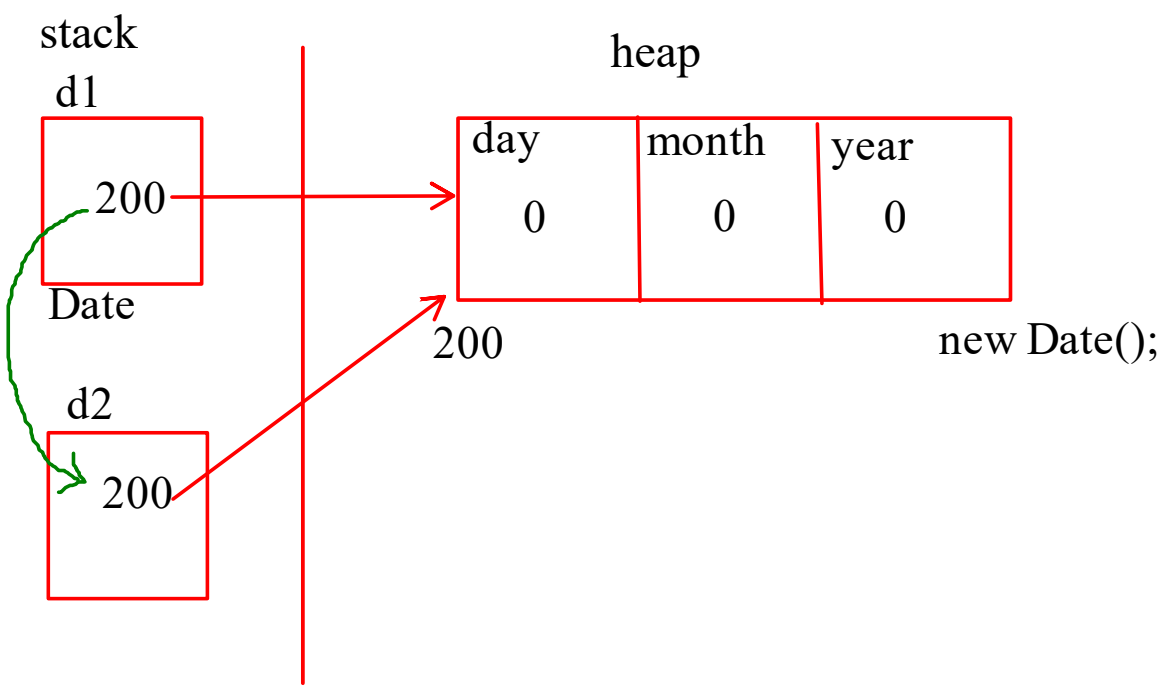
```
class Person{

void accept(){
}
}
class Employee extends Person{
@Override
void accept(){
super.accept()
}
}
```

```
super()
```

```
class Student extends Person{

}
```

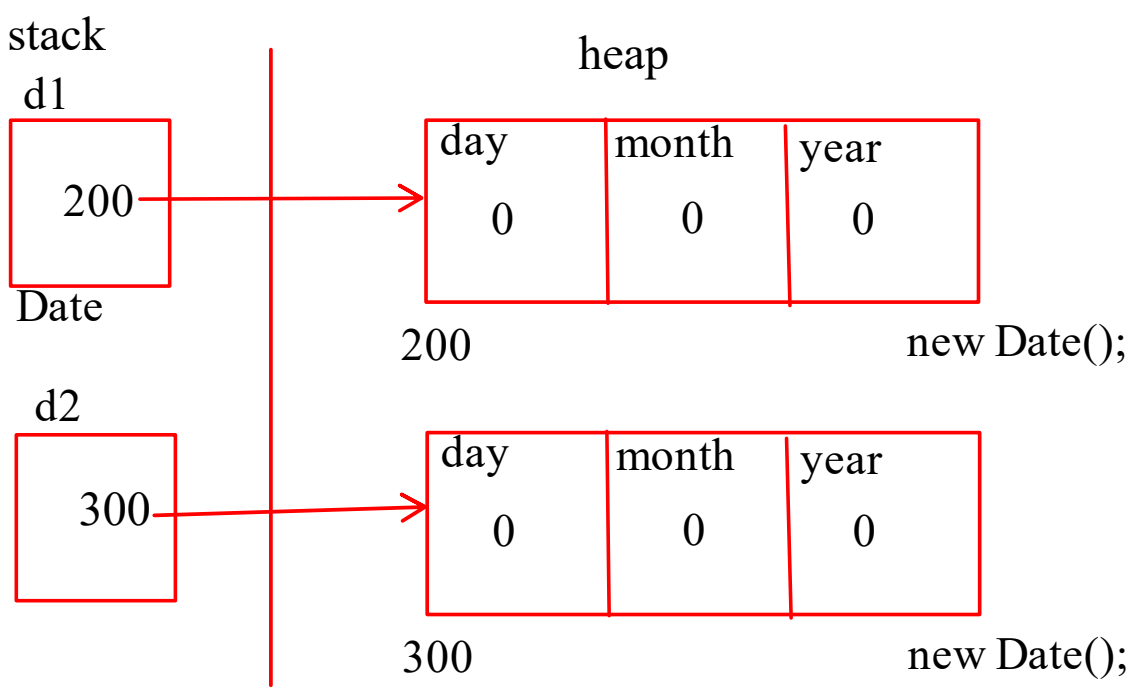
```
Person p; // reference
p = new Employee()
p = new Student();
if(p instanceof Employee)
Employee e = (Employee) p; // Downcasting
p.accept();
```

#Object class
- Super/root class of all the class hirerachy in java




```
Date d1 = new Date();
Date d2 = d1;
```

```
if(d1==d2)
```

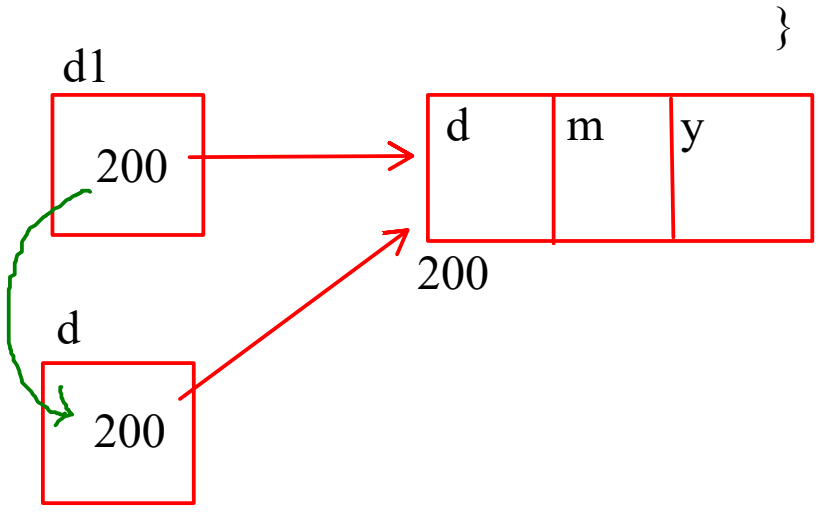


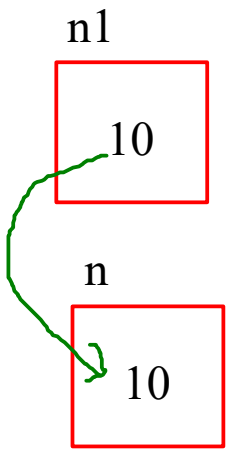

```
d1==d2
```

pass by value -> Primitive
pass by reference -> Non Primitive

```
void m1(int n){

}

void m2(Date d){

}
```

```
main(){
int n1 =10;
m1(n1);// pass by value
Date d1 = new Date(1,1,2000);
m2(d1);//pass by reference
}
```





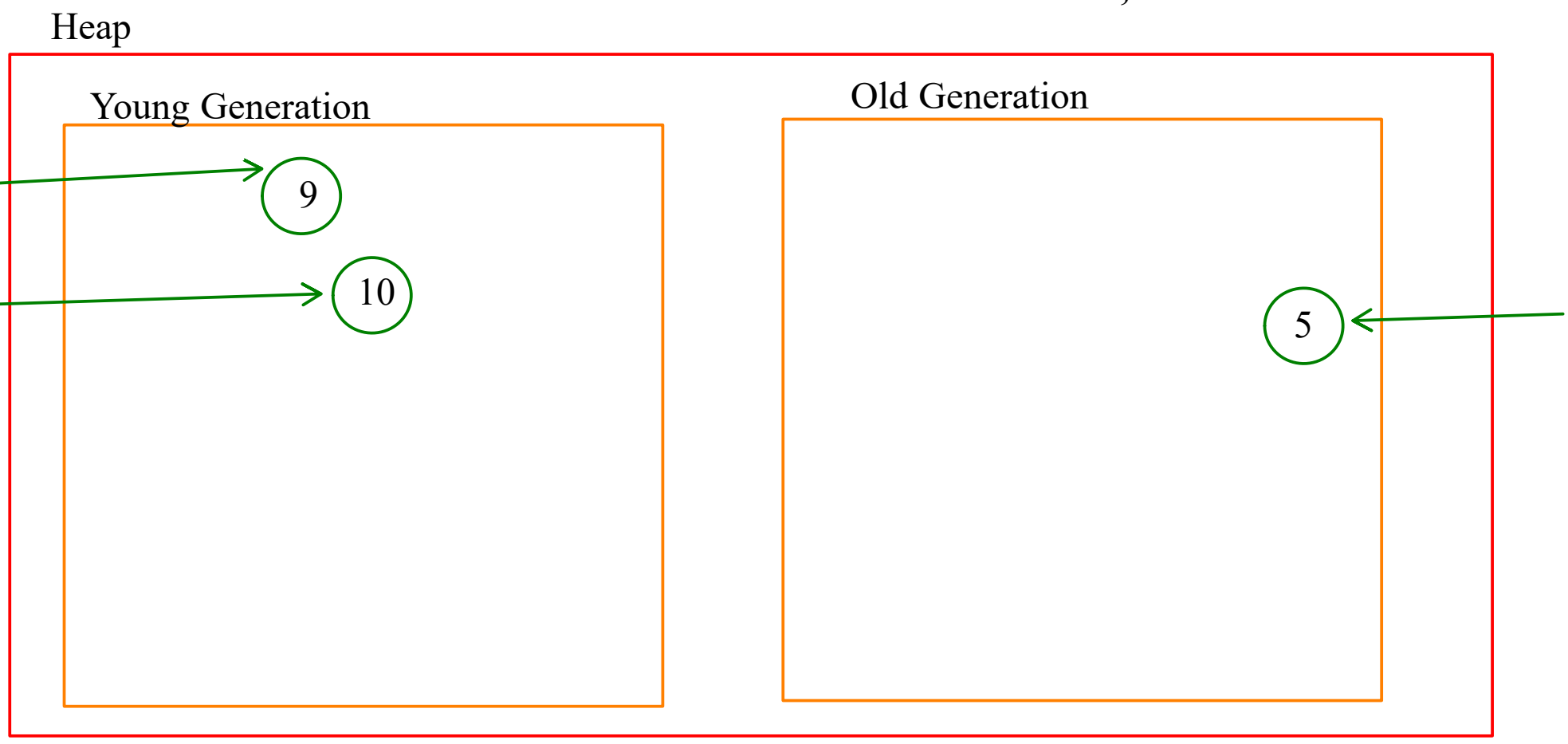

```
int arr[]={10,20,30};
for(int e : arr)

Employee arr[] = {new Employee(1,"Anil",10000),
                  new Employee(2,"Mukesh",20000)}
for(Employee e : arr)
```

Dynamic Method Dispatch

```
public static void main(){
        Date d1 = new Date(); // GC
        d1 = new Date();
}
```

```
public static void main(){
        Date d1 = new Date(); // GC
        d1 = null;
}
```

```
static void m1(){
        Date d1 = new Date(); // GC
}

public static void main(){
        m1();
}
```



Garbage Collection
Minor GC
Major GC

Mark and Compact Algorithm

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