Revision

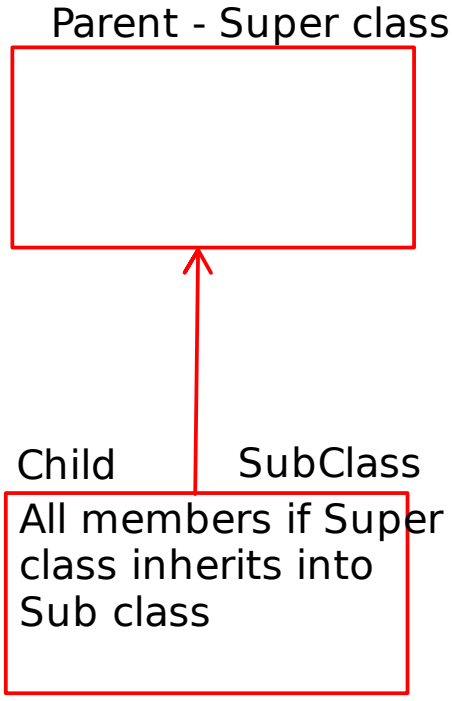
Hirerachy -> 1. Association (has-a relationship)
a. Aggegration
b. composition
2. Inheritance (is-a relationship)

Car has-a Engine // Composition
Car has-a InfotenmentSystem // Aggegration

```
class Engine{

}

class Car{
//Association
Engine e; // reference
InfotenmentSystem is;

{
// create the obj within class
e = new Engine(); // composition
}

}
```

```
class InfotenmentSystem{


}

main(){
InfotenmentSystem is;
// create the obj outside class
// Aggegration
is = new InfotenmentSystem();
}
```

Employee is-a Person
Manager is-a Employee




```
Parent p = new Parent(); // Parent()
Child c = new Child();// Parent()-->Child()
```

To call parameterized ctor of parent from child class use super keyword

If method overriding is done then to call the method of super class into the sub class we use super keyword

Method Overriding
- Redefining the method of super class into the sub class with same name and signature is called as Method overriding
- Reasons of Method Overriding
1. If implementation of super class method is partial complete
2. If implementation of super class method is 100% incomplete
3. If we require different implementation from the super class method.
- Rules of Method Overriding
1. Name and signature should be same
2. visibility should be same in sub class or it should be wider
3. Return type should be same or should be subtype
4. Exception ->

Final
1. Variable
2. Fields
3. Methods
4. class

```
class Person{

void accept(Scanner sc){

}
}
```

```
class Employee extends Person{

void accept(Scanner sc){
super.accept(sc);
}
}
```

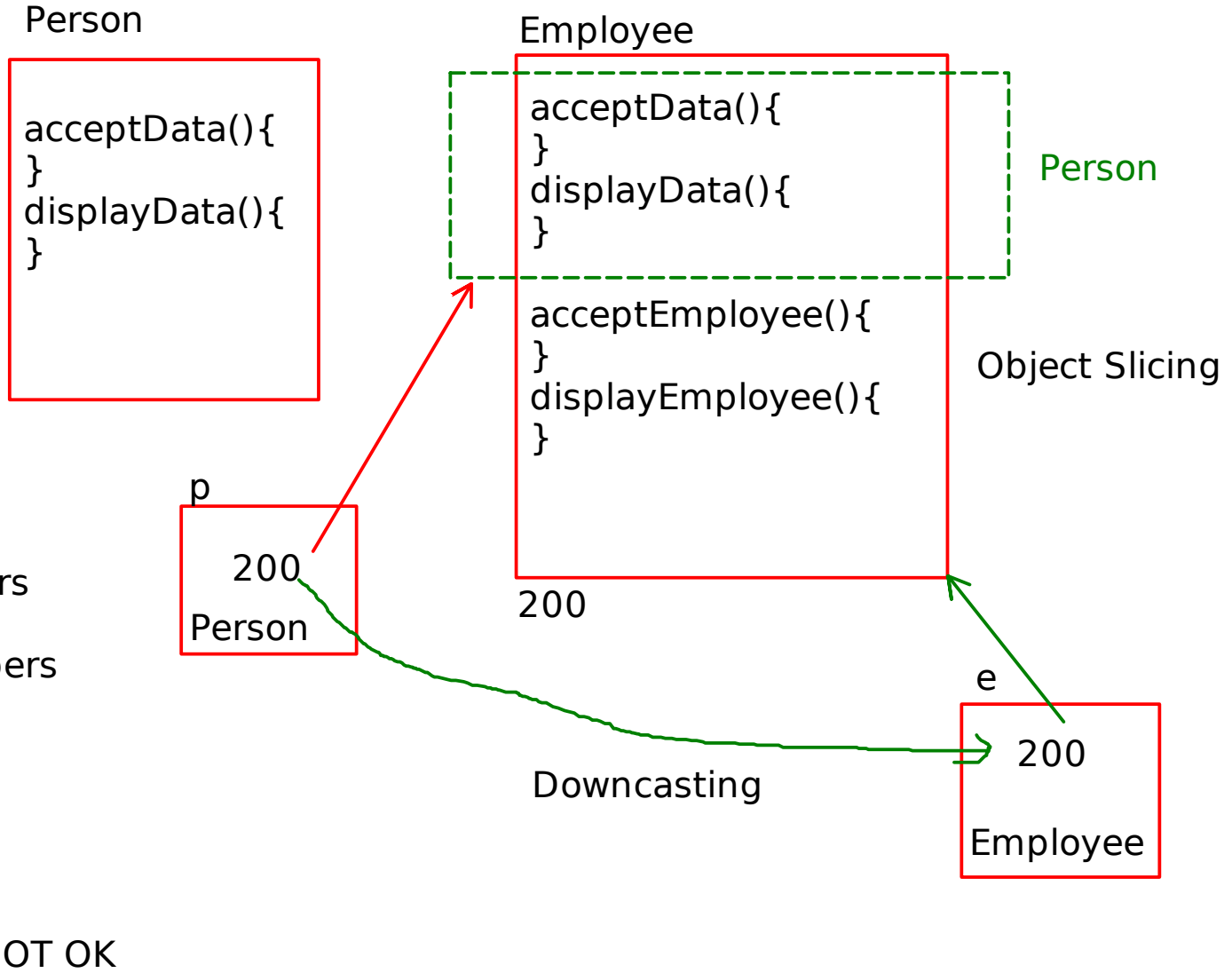
Upcasting -
Keeping the reference of sub class into the the super class reference is called as upcasting

Downcasting -
Converting the reference of super class into the sub class reference is called as Downcasting
At the time of downcasting explicit typecasting is mandatory
If downcasting fails java throws an exception ClassCastException

```
classs Person{
}

class Employee extends Person{
}

class Manager extends Employee{

}
```

```
Person person = new Person;

Employee employee = new Employee();

Manager manager = new Manager();

employee = manager; // upcasting
person = employee; // upcasting
```

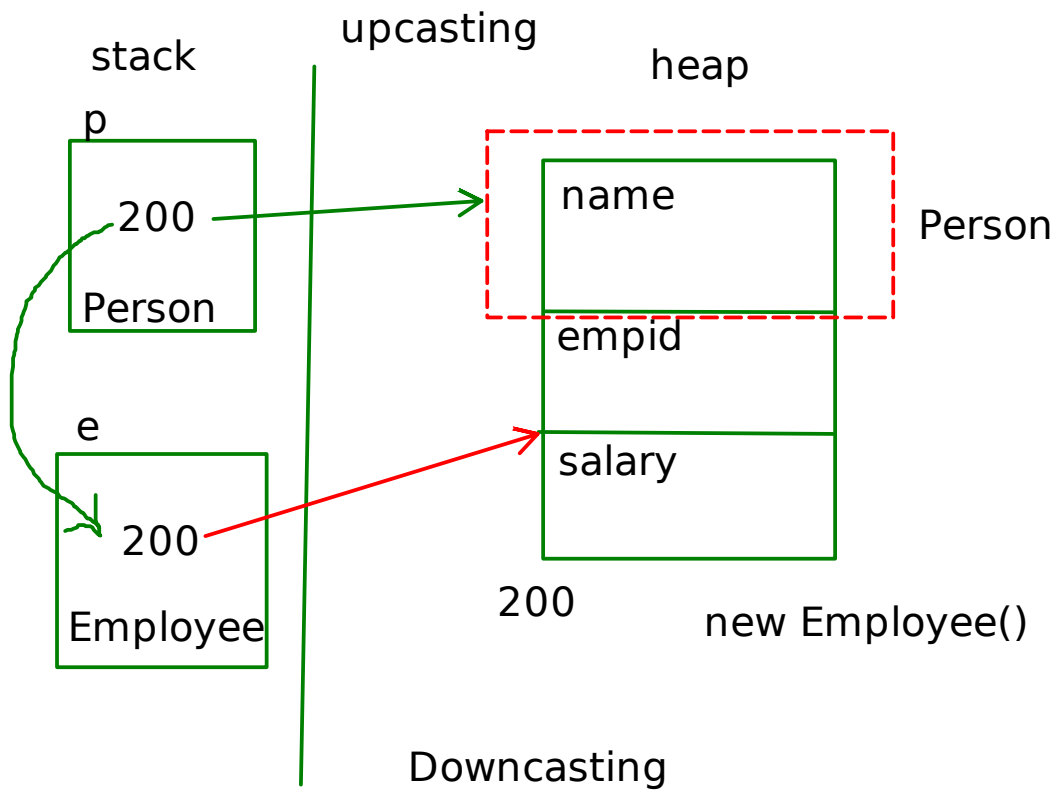
```
Person p;
p = new Employee();
p.acceptData();
p.displayData();
```

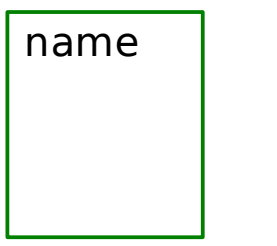


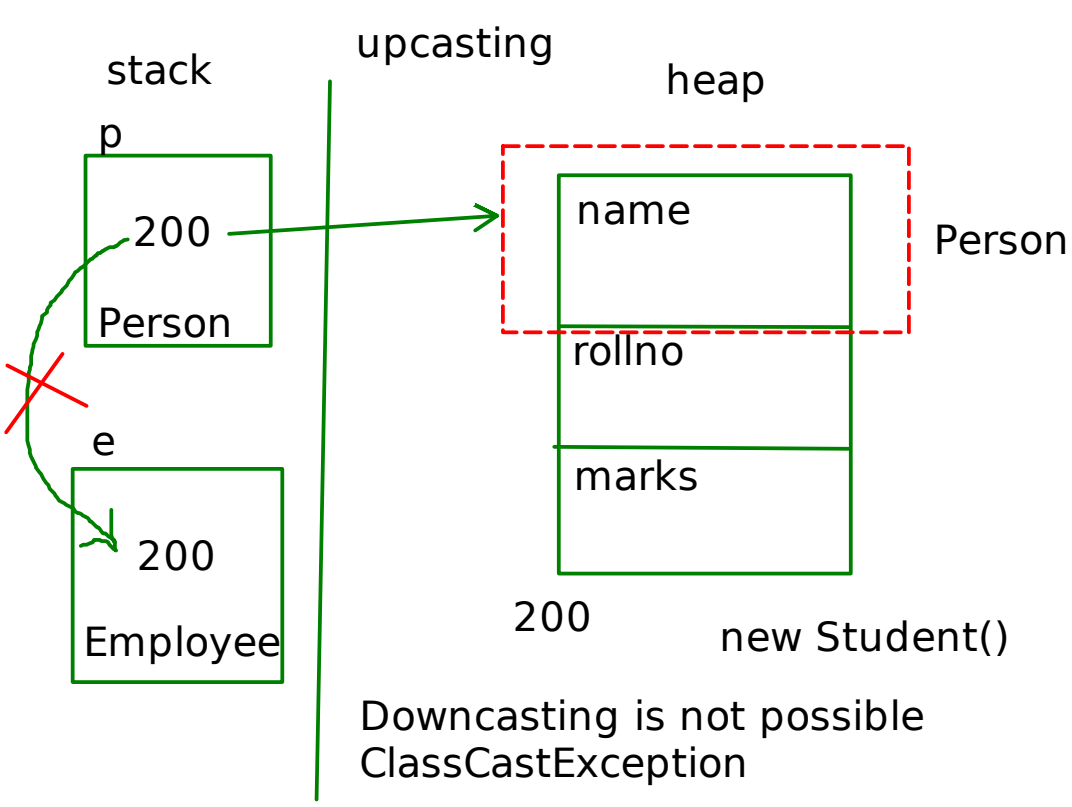
to point the members of the
super and sub class use
1. this ->
- to point at its own class members
2. super ->
- to point at its super class members

typeid -> instanceof

```
Person p = new Employee();

Employee e = new Person(); // NOT OK
```

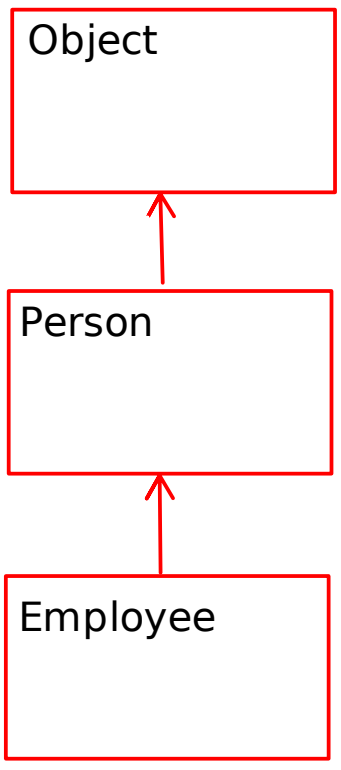






Object

point p1(2,3);
cout<<p1;




```
Object obj1 = new Person();
Object obj2 = new Employee();
```

Employee

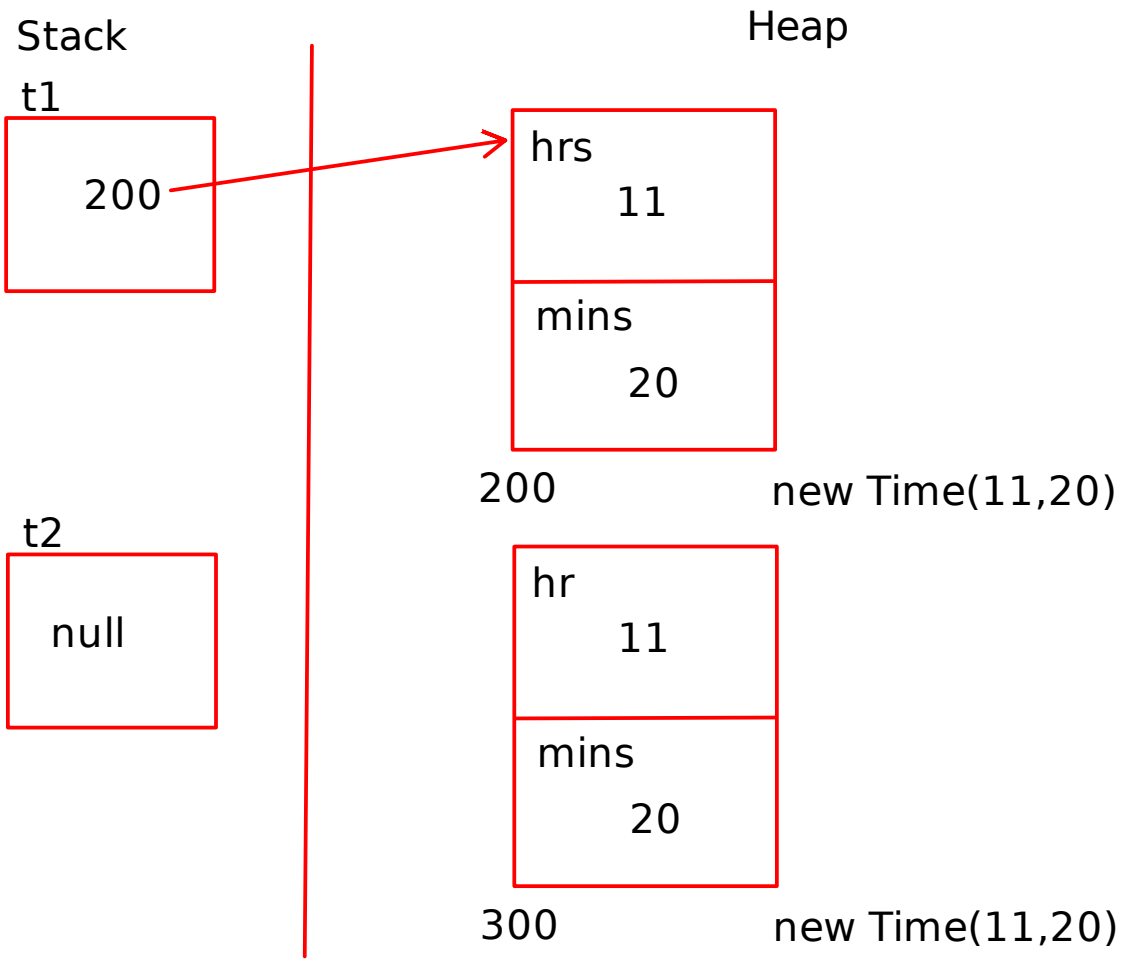
manager salesman

Salesmanager

```
Employee e = new SalesManager();
```

#toString() -> public String toString()

- If we want to display the state of an object in human readable format then we should override toString() of Object class
- toString method return the state of an object in String format.
- if we do not override the toStrig in our class then object class toString() gets called.
- Object class toString returns the state in the below format
  FullyQualifiedClassName @ Hashcode




```
Time t1 = new Time(11, 20);
Time t2 = null;

if(t1==t2)
if(200==300)

 if(t1.equals(null))

 t1->this, t2->obj
 boolean equals(Object obj){
 if(obj==null)
        return false;
 if(this==obj)
        return true;
 if(obj instance of Time){
 Time t2 = (Time) obj;
 if(this.hrs==t2.hrs &&
 ths.mins==t2.mins)
        return true;
 }

        return false;

 }
```