```
if(basket[i].getTaste().equals("sour"))
basket[i].setIsFrest(false);
```

```
Upcasting
Superclass super = new Superclass();

override
method1(); // super/sub

if(super instanceof Subclass)
        Subclass sub = (Subclass)super;
```

```
class Fruit
{
name
color
weight
isFresh = true

Fruit(String name)
{
this.name = name
}
}
```

```
class Apple{

Apple(){
super("Apple")
}

}
```

Object class
Super class of all the classes in java
11 methods in Object class
- toString()
    - to represent state of an object in human readable form
- equals()

```
 Fruit f1 = new Apple();
```

```
class Employee{
id
name
salary
toString()->id,name,sal
}

Employee e =  new Employee
sysout(e.toString());
// FQCL@hashcode
```

```
  @override
  boolean equals(Object obj){
        if(obj==null)
        return false;
        else if(this==obj)
        return true;
        else if(obj instanceof Emp)
        {
              Emp e = (Emp)obj;
              if(e.empid==this.empid.....)
              return true;
        }
  return false;
  }
```

```
e1 = new Emp();
e2 = new Emp();

e1.equals(e2)
```

Abstract
- abstract is a keyword in java
- we can make
1. method as abstract
2. class as abstract

Abstract Method
- If implementation of a method is 100% incomplete in super class then such methods should be decalred as abstract
- abstract methods do not have a body.
- abstact methods must be declared inside abstract class only.

Abstract Class
- If in a class we declare a method as abstract then the class by default becomes an abstract class.
- in java we have to declare such classes as abstract using the `abstract` access modifier
- For abstract class we cannot instantiate it, i.e object of abstract class cannot be created.
- we can only create the reference of the abstract class.
- abstract class can have abstract as well as non abstract methods

Fragile Base Class Problem

Interface -> Java 7

- It is a protocol that every subclass has to follow
- It is a specification for all the classes
- Interface is declared with the keyword `interface`
- interface consists of only abstract methods.
- we cannot define non abstact methods inside interface.
- Interface is the solution to your fragile base class problem
- interfaces are immutable i.e once declared it should not be modified.

```
class Employee{

}
```

```
interface Printable{

}
```

Abstract class

Employee

Manager

Salesman

```
Manager extends Employee
Salesman extends Employee

Employee e1 = new Manager();
Employee e2 =  new Salesman();


// upcasting
Accectable a1 = new Employee();
Accectable a1 = new Circle();
Accectable a1 = new Product();
```

Interface

```
interface Acceptable{
public abstract void acceptData(Scanner sc);
}
```

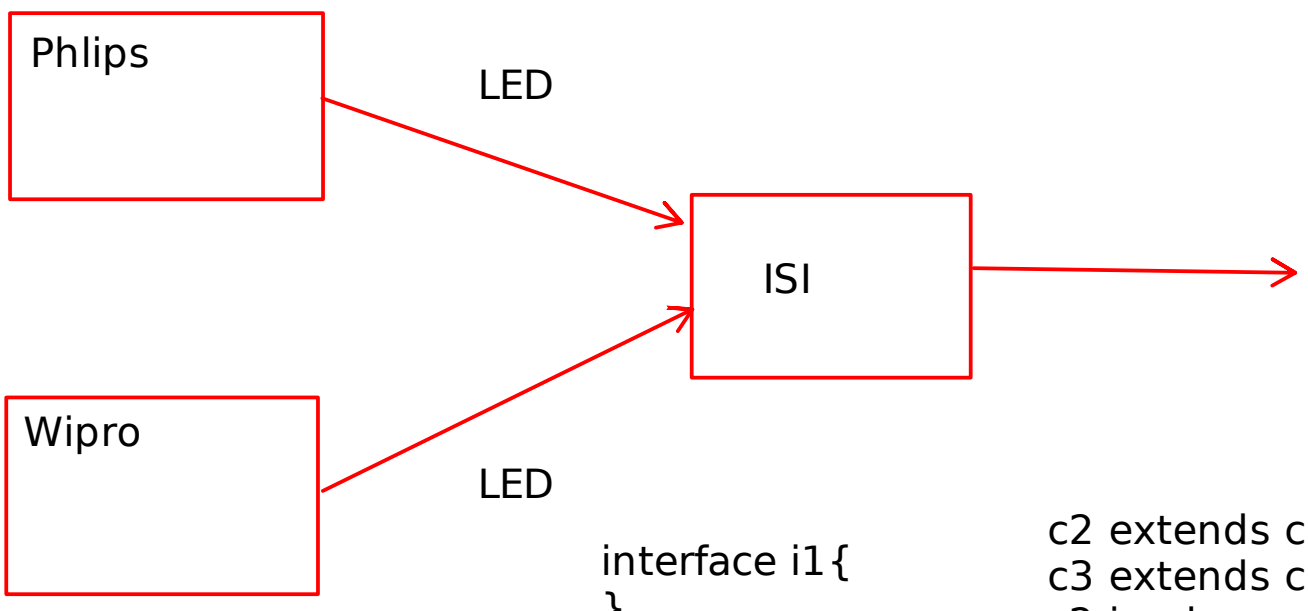
```
// following the specification
Employee implements Acceptable
{
override
acceptData(){

}

// following the specification
Circle impelments Acceptable
{
acceptData(){

}
}
Product implements Acceptable{
acceptData(){

}
}
```



Types of inheritance
1. Single Inheritance
2. Multilevel Inheritance
3. Multiple Inheritance
4. Hirerachihcal Inheritance
5. Hybrid Inheritance

```
interface i1{
}
interface i2{
}
interface i3{
}

class c1{
}
class c2{
}
class c3{
}
```

```
c2 extends c1 // OK
c3 extends c1,c2// NOT OK
c2 implements c1 // NOT OK

c2 implements i1 // OK
c2 implemets i1,i2 // OK
c2 extends i1 // NOT OK

i2 extends i1 // OK
i2 implements i1 // NOT OK
i3 extends i1,i2 // OK

class c2 extends c1 implements i1 // OK

class c3 extends c1 implements i1, i2 // OK
```

By default methods in interface are public and abstract
Interface cannot have the constructors
We can define the fields in the interface however those fields are public,static and final by default.

```
abstract Shape implements Acceptable{
area;
displayArea(){
sysout("Area =" +area)
}

abstract calculateArea();
}

Cirlce extends Shape{

acceptData(){
}
calculateArea(){
}

}

Rectangle{

}
```

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

Marker Interface
- An interface that is used to provide extra information / metadata to the JVM.
- An interface that do not have any methods, or an empty interface is called as Marker interface

```
Employee e1 = new Employee();
Employee e2 = new Employee();
```

orderdate
deliverydate
booking date
appointment date

Execption Handling

Error ->
Execption->

try, catch,throw