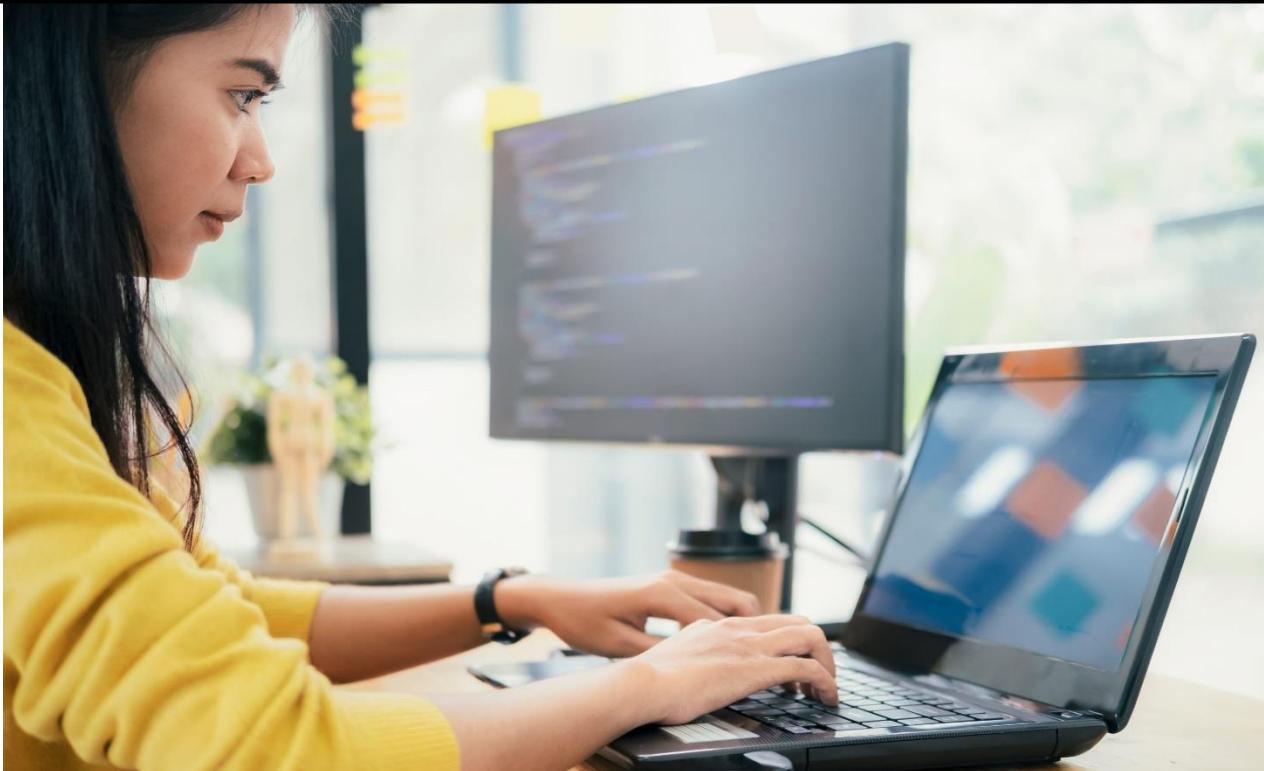




Build Your IT Skill

ផែវកំស្ថាល់ពី OOP

(Object Oriented Programming)

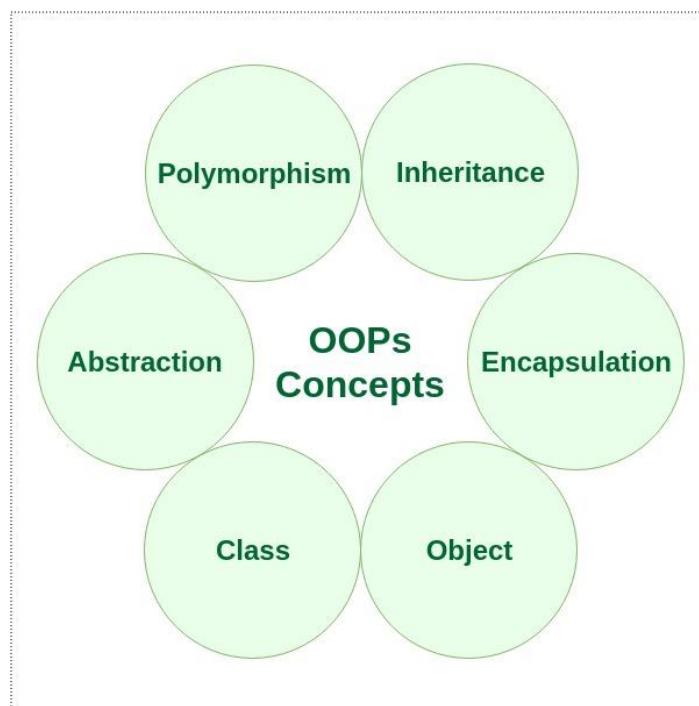


យោបាយនៃការងារ:
បច្ចេកទេសជិតអ៊ូរប៉ា
អនុបណ្ឌិត RUPP, TKU
(Software Engineering)

OOP Class /Object

I. ចូលរួមទៅលក្ខណៈ OOP?

OOP ឬរាយពិភ័យកុំប្រទែន ឬ Object Oriented Programming សំដោលីការបង្កើត Software ដោយពីនិងលើ ការប្រើប្រាស់នូវ Class និង Object ។ OOP ត្រូវបានគេប្រើប្រាស់យ៉ាង ពេញនិយមលើការបង្កើតនូវ Software ឬ project ជាមួយនឹង ក្រុមហុនដារាយ សម្រាប់មានការងារប្រើប្រាស់នូវ Concept OOP ដូចជា Java, C++, C#, Python, PHP, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp និង Smalltalk ។



Simula (ឆ្នាំ 1967) ត្រូវបានគេទទួលយកជាទូទៅបានការងារមួយដែលមានលក្ខណៈ:

មួយនឹង OOP (Object Oriented Programming) ។

វាត្រូវបានបង្កើតឡើងសម្រាប់បង្កើត Simulation

ដែលអ្និតដែលត្រូវបានគេហេចជា Object គឺជាតិ

ឯកសារបង្ហាញព័ត៌មាន ។



The Simula logo features the word "simula" in a bold, red, sans-serif font. The letters are vertically aligned and have horizontal stripes running through them, giving it a textured appearance. A small red circle is positioned above the letter "i".

OOP ឡូវការណ៍ Concept ទាំងអស់

- ១) Abstraction: គឺសំដែរលើការរៀបចំ Data និង Method តាមលក្ខណៈមួយតាមលំដាប់លំដោយ ដែលត្រូវបិទចាំនៃ ឬ ត្រូវបង្ហាញពីក្នុង Class ។
Encapsulation: គឺសំដែរលើការបញ្ចប់លក្ខណៈ Data និង Methods ដោយក្នុងការគ្រប់គ្រងមួយត្រូវ។
- ២). Inheritance: សំដែរលើការបង្កើតឡើង Class ដើម្បីមួយចំណែក Class ដែលមានស្រាប់ ឬ ការបង្កើត Class ដែលទទួលទិន្នន័យពីភ្នាក់។
- ៣). Polymorphism: សំដែរលើការប្រើប្រាស់នូវទំនួរប្រើប្រាន់ Object ទៅមួយ។

II. ជួលផ្តល់ដោលបញ្ជាដោយ Class ?

Class សំដែរលើការបញ្ចប់នូវ Data & Method ដាក់ដោយការគ្រប់គ្រងដោយ Keyword Class និង ក្រោមឈ្មោះតែមួយរួមគ្នា ឬ Class គឺជាពួកគៈគ្នាប់រួមមួយដើម្បីបង្កើត object ។
Data member: គឺជាប្រភេទ Variable ឬ Array ដែលសំរាប់ផ្ទុក ទិន្នន័យដោបេណ្ឌានេះ អាសន្ន។

ឧទាហរណ៍ ៩

```
1 //Data Member in class
2 import java.util.*;
3 class Student{
4     public int id;
5     public String name;
6     public String sex;
7     public float score;
8 }
```

Method member: គឺជាប្រភេទ Function ដែលបង្កើតក្នុងថាមពលនៃការអនុវត្តយ។
Method អាចបង្កើត Return function និង Non Return function អាស្រែយណើការអនុវត្តដោយ។

ឧទាហរណ៍ ៩

```
1 //Data Member in class
2 import java.util.*;
3 class Student{
4     public int id;
5     public String name;
6     public String sex;
7     public float score;
8     //Method Member
9     public void inputData()
10    {
11        Scanner objin=new Scanner(System.in);
12        System.out.print("Input ID=");
13        this.id=objin.nextInt();
14        System.out.print("Input Name=");
15        this.name=objin.nextLine();
16        System.out.print("Input Gender=");
17        this.sex=objin.next();
18        System.out.print("Input Score=");
19        this.score=objin.nextInt();
20    }
21 }
```

III. ចូលរួមទៅលើ Object ?

Object គឺជាបស់ដែលកែតែងឡើងពី Class និង ដែលមានលទ្ធភាពអាមេរិកស៊ី នូវទិន្នន័យបស់ Class ត្រូវបានតែងតាំងប្រភេទ Public តើបីណានេះ។ ដើម្បីបង្កើតទូទៅ object ត្រូវបង្កើតឡើងនៅក្នុង keyword new ដោយស្រប។

```
22 class TestStudents{
23     public static void main(String args[])
24    {
25        Student obj1=new Student();
26        //Or
27        Student obj2;
28        obj2=new Student();
29        obj1.inputData();
```

ឧទ្ធសាស្ត្រ ៩

```
2 import java.util.*;
3 class Student{
4     public int id;
5     public String name;
6     public String sex;
7     public float score;
8     //Method Member
9     public void inputData()
10    {
11        Scanner objin=new Scanner(System.in);
12        System.out.print("Input ID=");
13        this.id=objin.nextInt();
14        System.out.print("Input Name=");
15        this.name=objin.next();
16        System.out.print("Input Gender=");
17        this.sex=objin.next();
18        System.out.print("Input Score=");
19        this.score=objin.nextInt();
20    }
21    public void outputData()
22    {
23        System.out.print("\nID="+id);
24        System.out.print("\nName="+name);
25        System.out.print("\nGender="+sex);
26        System.out.print("\nScore="+score);
27    }
28}
29 class TestStudents{
30     public static void main(String args[])
31    {
32        //Create Object Class students
33        Student obj1=new Student();
34        obj1.inputData();
35        obj1.outputData();
36    }
37}
38}
```

ឧទាហរណ៍ ៤

```
1 import java.util.*;
2 public class Test{
3     private int x;
4     private int y;
5     private int c;
6     public void input()
7     {
8         Scanner objin=new Scanner(System.in);
9         System.out.print("Input Value X=");
10        x=objin.nextInt();
11        System.out.print("Input Value Y=");
12        y=objin.nextInt();
13        System.out.print("Input Value Z=");
14        z=objin.nextInt();
15    }
16    private int sum()
17    {
18        return x+y+z;
19    }
20    private int sub()
21    {
22        return x-y/z;
23    }
24    public void output()
25    {
26        System.out.print("X=" + x + " Y=" + y + " Z=" + z +"\n");
27        System.out.print("Sum=" + sum());
28        System.out.print("Sub=" + sub());
29    }
30 public class DemoTest{
31     public static void main(String args[])
32     {
33         Test obj=new Test();
34         obj.input();
35         obj.output();
36     }
37 }
```

IV. Constructor

Constructor គឺជាប្រភេទ function ពីសេសម្បយប្រភេទដែរធ្វើការដោយសំយប្រភេត្តនៅ

ពេល Object មួយច្បានកកៅតិថ្នីជូន។ Constructor មានពីរប្រភេទ ដូចខាង

១) Default Constructor: គឺជាប្រភេទ Constructor ដែលគេអាចបង្ការឡើង Default

អាយុទេរ Data member សំណើរក្សា Object កកៅតិថ្នីជូន។

២) Constructor With Parameter: គឺជាប្រភេទ Constructor ដែលអាចបង្ការឡើងពី
ខាងក្រោមរាយ៖ Object របស់ Class ។

```
1 import java.util.*;
2 public class Test{
3     private int x;
4     private int y;
5     private int z;
6     //Default Constructor
7     public Test()
8     {
9         this.x=0;
10        this.y=0;
11        this.z=0;
12    }
13    //Constructor with Parameter
14    public Test(int a,int b,int c)
15    {
16        this.x=a;
17        this.y=b;
18        this.z=c;
19    }
20    private int sum()
21    {
22        return x+y+z;
23    }
24    private int sub()
25    {
26        return x-y/z;
27    }
28    public void output()
29    {
30        System.out.print("X=" + x + " Y=" + y + " Z=" + z +"\n");
31        System.out.print("Sum=" + sum());
32        System.out.print("Sub=" + sub());
33    }
34 public class DemoTest{
35     public static void main(String args[])
36     {
37         //Using Default Constructor
38         Test obj;
39         obj=new Test();
40         obj.output();
41         //Constructor with parameter
42         obj=new Test(12,30,3);
43         obj.output();
44     }
45 }
```

សំណង់អនុវត្តន៍

- 1). ចូរបារើនីតិ Class មួយដែលមានលក្ខណៈ Employee ដែលផ្តល់ការពារជាប្រព័ន្ធឌីជីថទ្ទី Data member ពារជាប្រព័ន្ធឌីជីថទ្ទី id(String), Name(String), Sex(String) & Salary(double) និង មាន function ចំណួន ២ គឺ void Input() និង void Output() ដោយប្រើប្រាស់នូវ Console Application បន្ទាប់មកបង្កើតនូវ Object ចំណួនពីរ គឺ objemp1 និង objemp2 ដែលប្រើប្រាស់នូវ Class នេះបាន?

V. Object Array

វាតីប្រភេទ Object ដែលរាយចកចង្វារទិន្នន័យលើខ្លួនវា ប្រាក់ចិន ដែលខិត្តពី Object ដម្លាត់។

```
public class DemoTest{
    public static void main(String args[])
    {
        Students []stu=new Students[20];
        for(i=0;i<20;i++)
        {
            stu[i]=new Students();
        }
    }
};
```

ឧទាហរណ៍ទី១ ទៅ ចូរធ្វើការបង្កើតនូវ Class មួយលក្ខណៈ Books បន្ទាប់មកបង្កើតនូវ Data member Code(int), title(String), subject(String) និង price(float) និង Constructor ពីរគឺ Books(), Books(_,_,_,_). និង function ពីរគឺ void input(), void output() និង property set_(), get_()?



“តាមីនាពន្លំប្រាកប្រើប្រាស់បណ្តិ៍បង្កើតបន្ទាត់
ប្រាកប្រាស់នៃបង្កើតបន្ទាត់
និងតាមីកប្រើប្រាស់...!”

—ក្រុមរោងធមិត្តនុ

```
import java.util.*;
public class Books {
    private int code;
    private String title;
    private String subject;
    private float price;
    //Default Constructor
    public Books()
    {
        this.code=0;
        this.title="N/A";
        this.subject="N/A";
        this.price=0;
    }
    //Constructor with Parameter
    public Books(int i, String n, String s, float p)
    {
        this.code=i;
        this.title=n;
        this.subject=s;
        this.price=p;
    }
    //Property get
    public int getCode()
    {
        return code;
    }
    //property set
    public void setTitle(String n)
    {
        this.title=n;
    }
    public void setSubject(String s)
    {
        this.subject=s;
    }
    public void Price(float p)
    {
        this.price=p;
    }
    public void input()
    {
        Scanner objin=new Scanner(System.in);
        System.out.println ("Input Code=");
        code=objin.nextInt();
        System.out.println ("Input Title=");
        title=objin.next();
        System.out.println ("Input Subject=");
        subject=objin.next();
        System.out.println ("Input price=");
        price=objin.nextFloat();
    }
    public void Output()
    {
        System.out.println ("Code=" + code);
        System.out.println ("Title=" + title);
        System.out.println ("Subject=" + subject);
        System.out.println ("Price=" + price);
    }
}
```

ឧបាទរណី៖ ចូរបង្កើតនូវ Object ជាលក្ខណៈ Array បន្ទាប់មកបង្កើត function ចំនួន ពីតែ Search(), Update() និង Sort()?

```
import java.util.*;
public class TestBook {
    public int Search(Books book[],int n)
    {   Scanner objin=new Scanner(System.in);
        int scode,i;
        int index=-1;
        int b=0;
        System.out.println ("Input Book Code to search=");
        scode=objin.nextInt();
        for(i=0;i<n;i++)
        {
            if(scode==book[i].getCode())
                book[i].Output();
                index=i;
                b=1;
                break;
        }
        if(b==0) System.out.println ("Search not found");
        return index;
    }
    public void Update(Books book[],int n)
    {   Scanner objin=new Scanner(System.in);
        int index=Search(book,n);
        String ntitle,nsub;
        float nprice;
        if(index!=-1)
        {

            System.out.println ("Input New Title=");
            ntitle=objin.next();
            System.out.println ("Input New Subject=");
            nsub=objin.next();
            System.out.println ("Input New price=");
            nprice=objin.nextFloat();
            book[index].setTitle(ntitle);
            book[index].setSubject(nsub);
            book[index].setPrice(nprice);
            System.out.println ("Update Completed.....!");
        }
        else{
            System.out.println ("Invalid Updated.....!");
        }
    }
}
```

```
public TestBook() {  
    Scanner objin=new Scanner(System.in);  
    int op,n=0;  
    String st;  
    Books []book=new Books[20];  
    do{  
  
        System.out.println ("1. Input");  
        System.out.println ("2. Output");  
        System.out.println ("3. Search");  
        System.out.println ("4. Update");  
        System.out.print ("Choose One=");  
        op=objin.nextInt();  
        switch(op)  
        {  
            case 1:{  
                System.out.print ("Input Number Array=");  
                n=objin.nextInt();  
                for(int i=0;i<n;i++)  
                {  
                    book[i]=new Books();  
                    book[i].input();  
                }  
            }break;  
            case 2:{  
                for(int i=0;i<n;i++)  
                {  
                    book[i].Output();  
                }  
            }break;  
            case 3:{  
                Search(book,n);  
            }break;  
            case 4:{  
                Update(book,n);  
            }break;  
        }  
  
        System.out.println ("Press yes to Continue....!");  
        st=objin.next();  
    }while(st.equals("yes"));  
  
}  
public static void main (String[] args) {  
    new TestBook();  
}  
}
```

L

សំណង់អនុវត្តន៍

១). ចូរបង្កើត Class មួយដែលមានលក្ខណៈ Employee ដែលផ្តល់ការពារ Data member ៣ គីឡូ ឬ id(String), Name(String), Sex(String) & Salary(double) និងមាន function ចំណួន ២ គីឡូ void Input() និង void Output() និង Constructor ចំណួនពីរ ទៀតគីឡូ Employee(), Employee(,), Employee(,,,,)

ដោយប្រើប្រាស់នូវ Console Application បន្ទាប់មកបង្កើតនូវ Object ចំណួនពីរ គីឡូ objemp1 និង objemp2 ដែលប្រើប្រាស់នូវ Class នៅពានបន្ទាប់មកបង្កើតនូវ Object ArrayList?



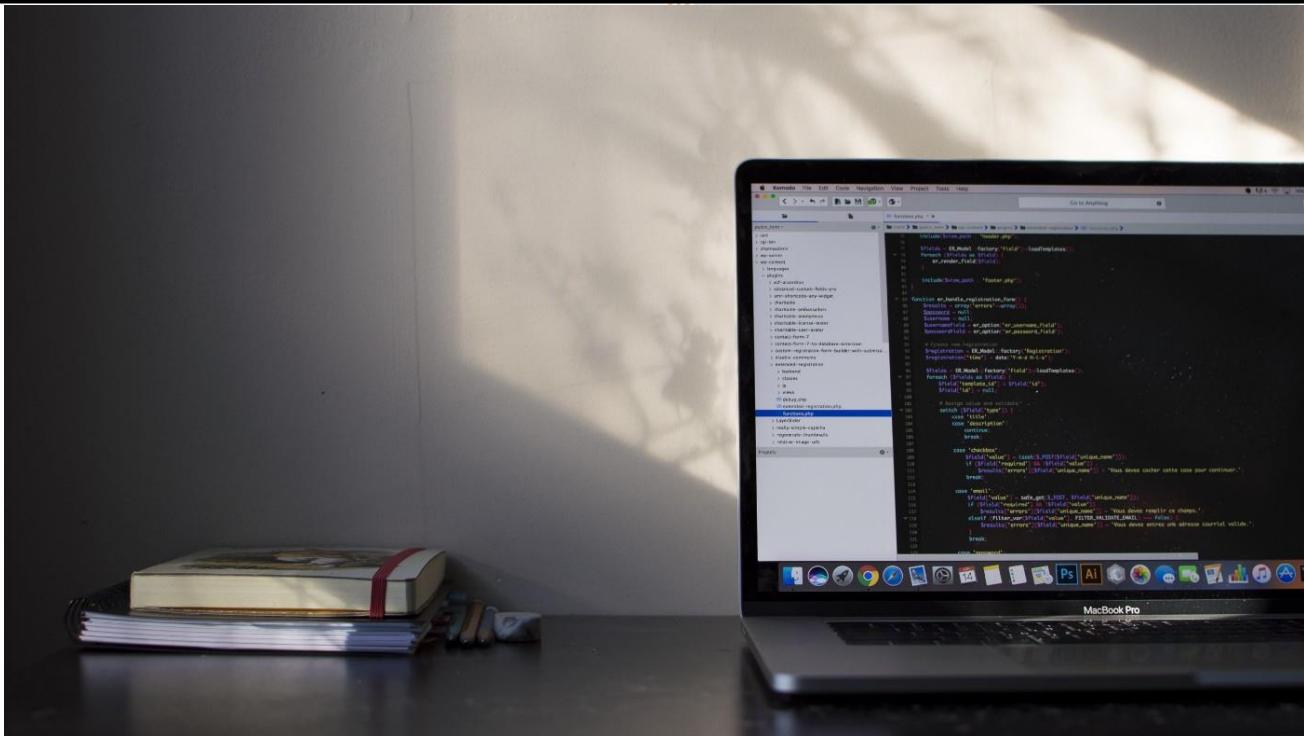
“ OOP តែងតាំងទៅជាប្លូនិលប្បុប្បន្នប្រព័ន្ធដើម្បី
នាយកដែលបានបង្កើតនូវ Framework ដើម្បីជួយបង្កើតគោលរាល់ Pattern
បានបង្កើតឡើង ហើយប្រព័ន្ធដើម្បីជួយបង្កើតគោលរាល់គឺជាមួយប្រព័ន្ធដើម្បី
ជួយបង្កើតគោលរាល់គឺជាមួយប្រព័ន្ធដើម្បី...!”

— វឌ្ឍនោយបិទិត្តន៍ —



Build Your IT Skill

ណែនាំស្ថាល់ពី Inheritance



រៀបចាននៅយោះ
ក្រុមហ៊ុនិតិផល
អនុបណ្ឌិត RUPP, TKU
(Software Engineering)

Inheritance

I. ឯកទេសចរណ៍បែងចែង Inheritance ?

Inheritance គឺជាជំណើរនៃការកែតែ Class ដើម្បីយកពេញពី Class ដែលមានស្រាប់បុគ្គលិក ឬការបង្កើតទូទៅ Relationship Class(ទំនាក់ទំនើស Class)។ Class ដែលមានស្រាប់គោរព Base Class Or Super Class និង class ដែលមានស្រាប់គោរព Derived Class ឬ Sub Class។
លក្ខណៈសំខាន់របស់ Inheritance គឺកាត់បន្ទូយនូវការសរសេរក្នុងបុគ្គលិកប្រកាសអញ្ចាតិ ដួងលាយ ក្នុងឱ្យការទំនាក់ទំនើស នៃ Class ធ្វើការចូរសំណាត់។

ទីនំនូវផែនការ

```
1 class derived-class extends Super-class
2 {
3     //methods and fields
4     .....
5 }
```

ឧបាទរណ៍ 1: គោរព Super Class មួយដែលផ្តល់ក្នុងទីនំនូវផែនការ 3 គីឡូ ឬ x,y,z

```
Start Page Test1.java * x
import java.util.*;
class MySuper{
    protected int x;
    protected int y;
    protected int z;
}
```

ឧទាហរណ៍ 2: ចូរបង្កើតឡើវ Sub Class ដែលអាចទាញទិន្នន័យពី Super Class ដែលមានលក្ខណៈ ជា Protected ឬ Public ត្រូវទាំងបង្កើតឡើវ object ដែលអាចប្រើប្រាស់ឡើវ Sub Class នៅៗបាន?

```
Start Page Test1.java * x
import java.util.*;
class MySuper{
    protected int x;
    protected int y;
    protected int z;
}
class MySub extends MySuper{
    private int a;
    private int sum()
    {
        return x+y+z+a;
    }
    private int sub()
    {
        return x-y+z;
    }
    public void Input()
    {
        Scanner objin=new Scanner(System.in);
        System.out.print("Input X=");
        x=objin.nextInt();
        System.out.print("Input Y=");
        y=objin.nextInt();
        System.out.print("Input Z=");
        z=objin.nextInt();
        System.out.print("Input A=");
        a=objin.nextInt();
    }
    public void Output()
    {
        System.out.print("X=" + x + "\n");
        System.out.print("Y=" + y + "\n");
        System.out.print("Z=" + z + "\n");
        System.out.print("A=" + a + "\n");
        System.out.print("Sum=" + sum() + "\n");
        System.out.print("Sub=" + sub() + "\n");
    }
}
class Test1{
public static void main (String[] args) {
    MySub obj=new MySub();
    obj.Input();
    obj.Output();
}
}
```

លំហាត់អនុវត្តន៍១

ចូរបង្កើតនូវ Super Class មួយណ៍៖ Person ដែលមាននូវ Data member ៥ គឺ Id(String), Name(String), Sex(String), Address(String) និង DOB(String) បន្ទាប់មកបង្កើតនូវ Function ចំណួនពីរ គឺ void Input() និង void Output()

លំហាត់អនុវត្តន៍២

ចូរបង្កើតនូវ Class Employee ដែលមាន data member 3 គឺ Salary(float), Rate(float) និង hour(int) បន្ទាប់មកបង្កើតនូវ Function ចំណួនពីរគឺ void Input(), void ouput() ដែល Function ទាំងពីរនេះ ត្រូវការពិនិត្យថាមួយចោរពី Class person មកប្រើប្រាស់បានៗម?



“ព្រាយុមាណជាក់បិត្តលើវានោយអស់បច្ចុប្បន្ន
មិនមែន មែន មែន , មិនមែន មិនមែន មិនមែន
មិនមែន នៅ ព្រាយុមាណយល់នូវលំដាប់
មិនមែន មិនមែន”

—ត្រូវឃាយជិត្តនូវ

II. ដើម្បីលោកស្រើ Function តិ៍ Super Class ត្រូវប្រើប្រាស់ឡើង keyword super.Input() និង super.Output().

```
class MySub extends MySuper{
    private int a;
    public void Input()
    {
        Scanner objin=new Scanner(System.in);
        //Calling Input from Super Class
        super.Input();
        System.out.print("Input A=");
        a=objin.nextInt();
    }
    public void Output()
    {
        //Calling Output from Super Class
        super.Output();
        System.out.print("A=" + a + "\n");
        System.out.print("Sum=" + sum() + "\n");
        System.out.print("Sub=" + sub() + "\n");
    }
}
```

III. ເນື້ອງ ໂຫວາ ຄູ່ຈຳ Constructor ຕີ Super Class

```
Start Page Test1.java * x
import java.util.*;
class MySuper{
    protected int x;
    protected int y;
    protected int z;
    //Default Constructor
    public MySuper()
    {
        x=0;
        y=1;
        z=2;
    }
    //Constructor with Parameter
    public MySuper(int x,int y,int z)
    {
        this.x=x;
        this.y=y;
        this.z=z;
    }
}
class MySub extends MySuper{
    private int a;
    public MySub()
    {
        //calling Default Constructor
        super();
        a=4;
    }
    public MySub(int a,int b,int c,int d)
    {
        //calling Constructor with parameter
        super(a,b,c);
        this.a=d;
    }
}
```

លំនាច់អនុវត្តន៍ ៣

តើលំហាត់អនុវត្តន៍ទី ១ ចូរបន្ទាន់មួយ Default Constructor និង Constructor With Parameter គឺដឹង
Super Class បន្ទាប់មកបង្កើតនៅ Sub class ហើយ Constructor ទាំងនេះចូលទៅប្រើប្រាស់បន្ទាម ?

ប្រព័ន្ធឌែល Inheritance

នៅក្នុង Inheritance ត្រូវបានគេបើនិច្ចការ ២ ទំនើសដូចជា៖

១) Single Inheritance: គឺជាប្រភេទ Inheritance ដែលមាន Super Class មួយអាច មាន
នូវ Sub Class ១ ទៅឡាយកទិន្នន័យបន្ទាត់។

២) Multiple Level inheritance : គឺជាប្រភេទ inheritance ដែលអាចមាន Super class
មួយ និង Sub class ២ ឡើងទៅដែលត្រូវទាញទិន្នន័យបន្ទាត់ Class គ្មាន។

៣) Hierarchical Inheritance: គឺជាប្រភេទ Inheritance ដែលមាន Super Class មួយ
ហើយមាន Sub Class ប្រើនទាញទិន្នន័យដោយផ្តាល់ពី Super Class ទាំងនេះ។



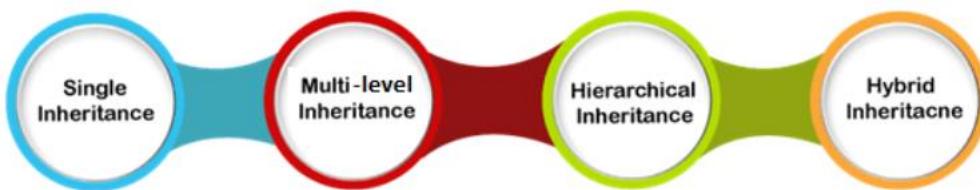
“**នើរិនុវត្តន៍ ឬ នូវរាយការណ៍នៃអនុវត្តន៍**
CODE សិន នើរិនុវត្តន៍ ឬ នូវរាយការណ៍នៃអនុវត្តន៍
នាករឡាចូល...!”

—ត្រូវរាយការណ៍នៃអនុវត្តន៍

Java supports the following four types of inheritance:

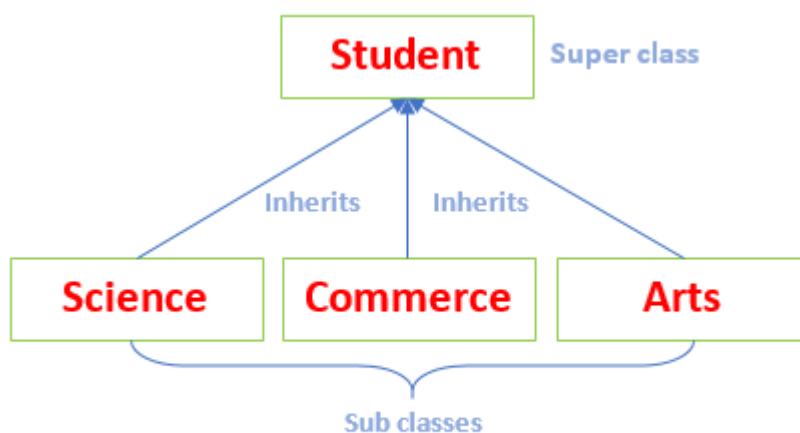
- Single Inheritance
- Multi-level Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

Types of Inheritance in Java



Note: Multiple inheritance is not supported in Java.

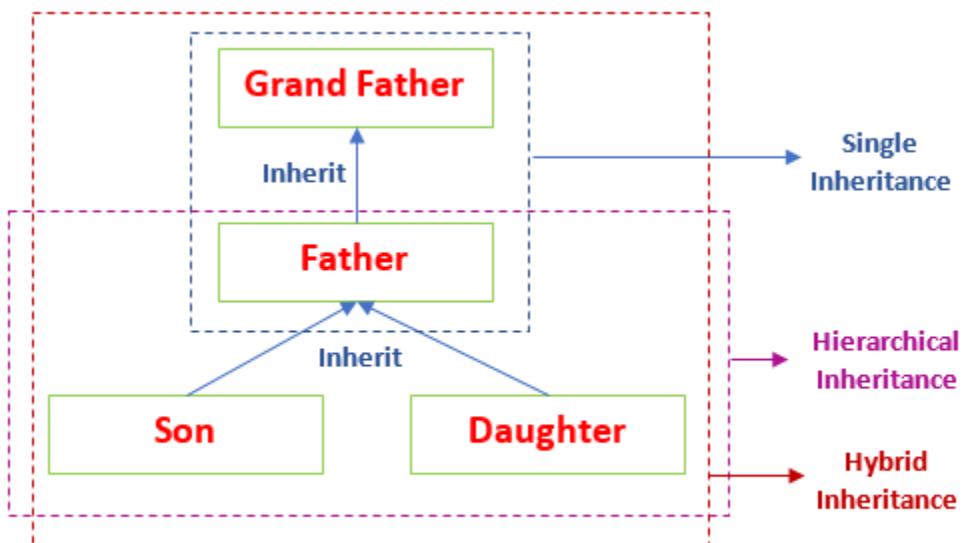
ឧទានការណ៍ Hierarchical Inheritance



Hierarchical Inheritance

```
1 //Super class
2 class Student
3 {
4     public void methodStudent()
5     {
6         System.out.println("The method of the class Student invoked.");
7     }
8 }
9 class Science extends Student
10 {
11     public void methodScience()
12     {
13         System.out.println("The method of the class Science invoked.");
14     }
15 }
16 class Commerce extends Student
17 {
18     public void methodCommerce()
19     {
20         System.out.println("The method of the class Commerce invoked.");
21     }
22 }
23 class Arts extends Student
24 {
25     public void methodArts()
26     {
27         System.out.println("The method of the class Arts invoked.");
28     }
29 }
30
31 public class HierarchicalInheritanceExample
32 {
33     public static void main(String args[])
34     {
35         Science sci = new Science();
36         Commerce comm = new Commerce();
37         Arts art = new Arts();
38 //all the sub classes can access the method of super class
39         sci.methodStudent();
40         comm.methodStudent();
41         art.methodStudent();
42     }
43 }
```

ඉංජායෙන්ස් Hybrid Inheritance



Hybrid Inheritance

```
1 //parent class
2 class GrandFather
3 {
4     public void show()
5     {
6         System.out.println("I am grandfather.");
7     }
8 }
9 //inherits GrandFather properties
10 class Father extends GrandFather
11 {
12     public void show()
13     {
14         System.out.println("I am father.");
15     }
16 }
17 //inherits Father properties
18 class Son extends Father
19 {
20     public void show()
21     {
22         System.out.println("I am son.");
23     }
24 }
25 //inherits Father properties
26 public class Daughter extends Father
27 {
28     public void show()
29     {
30         System.out.println("I am a daughter.");
31     }
32     public static void main(String args[])
33     {
34         Daughter obj = new Daughter();
35         obj.show();
36     }
37 }
```

សំណង់នូវក្នុង

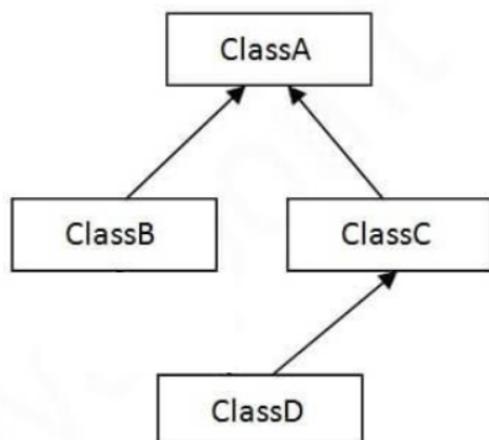
ប្របដើម្បីត្រួវ Super Class មួយឈោះ Person ដែលមានទីន៍ Data member គឺ Id(String), Name(String), Sex(String), Address(String) និង DOB(String) និង បង្កើតនូវ Constructor ចំណួនពីរឡើងគឺ Person(), Person(_____,_____,_____,_____) បន្ទាប់មកបង្កើតនូវ Function ចំណួនពីរ គឺ void Input() និង void Output()

ប្របដើម្បីត្រួវ Sub Class មួយគឺ Students ដែលមាន score 5 មុខវិធាន, Constructor ចំណួនពីរ និង Function Member ចំណួនពីរ ដែលមិនអាចប្រើបាយបាន

ប្របដើម្បីត្រួវ Sub Class មួយ ឡើងគឺ Employee ដែលមាន Data member ជាពាមាម (float) និង Constructor ចំណួនពីរ និង Function Member ចំណួនពីរ ដែលមិនអាចប្រើបាយបាន

បុង្កាយចូលរោង Sub Class ទាំងពីរគ្មានយកទិន្នន័យពី Super Class មួយនេះ? និងចូលបង្កើត Class Teacher ដែលមាន Data member ពីរ គឺ Subject និង RoomNo និង Constructor ចំណួនពីរ និង Function Member ចំណួនពីរ ដែលមិនអាចប្រើបាយបាន

ម៉ោងទាញយកទិន្នន័យពី Class Students





Build Your IT Skill

ណែនាំស្ថាល់ពី Polymorphism



យោបាយនៃយោង
កម្រិតបច្ចុប្បន្ន
អនុបណ្ឌិត RUPP, TKU
(Software
Engineering)

Polymorphism

I. ចូលរួមនៅ Polymorphism ?

ពាក្យបា Poly + morphism គឺសំដែរលើការប្រើប្រាស់ទំនើតប្រើប្រាស់ Object Super Class និង លក្ខណៈរបស់ Methods បានប្រើប្រាស់ទំនើតក្នុង Super Class និង Sub Class នៅក្នុងបំនុចនេះអ្នកនិងសិក្សាលើ

1. Method Overloading

2. Method Overriding

3. Types of Polymorphism – Runtime and compile time

1. Method Overloading: សំដែរលើការបង្កើតឡើយ Methods ដែលមានលក្ខណៈ ដូចត្រូវ បានបង្កើតឡើយតាមការតាមតម្លៃស្ថាលើ Return type function និង បំនុយ parameter របស់ function ទាំងនេះ។

ឧទាហរណ៍១ Overload ឬស្ថាល្អតម្លៃនៃ Parameter

```
1 class Adder{  
2     static int add(int a,int b){return a+b;}  
3     static int add(int a,int b,int c){return a+b+c;}  
4 }  
5 class TestOverloading1{  
6     public static void main(String[] args){  
7         System.out.println(Adder.add(11,11));  
8         System.out.println(Adder.add(11,11,11));  
9     }  
10 }
```

ଓভেରলୋଡିଂ Overloading ଅପଣାଗ୍ରହିତ ପ୍ରାକ୍ରିଯାଷ୍ଟେ

```
1 class Adder{  
2     static int add(int a, int b){return a+b;}  
3     static double add(double a, double b){return a+b;}  
4 }  
5 class TestOverloading2{  
6     public static void main(String[] args){  
7         System.out.println(Adder.add(11,11));  
8         System.out.println(Adder.add(12.3,12.6));  
9     }  
10 }
```

- Constructor Overloading: សំដែកលើការបង្កើតឡើវា Constructor ដែលមាន
យោង: ផ្ទចត្តា បាប់ពីរឡើងទៅតើមានភាពខ្លួនគ្នាលើ ចំណួន parameter
របស់ Constructor ទាំងនេះ។

```
Start Page Employee.java * X
public class Employee {
    private String name;
    private String address;
    private int number;
    public Employee(String n, String a, int num)
    {
        name=n;
        address=a;
        number=num;
    }
    public Employee()
    {
        name="N/A";
        address="N/A";
        number=0;
    }
    public Employee(int num)
    {
        name="N/A";
        address="N/A";
        number=num;
    }
    public Employee(String n, int num)
    {
        name=n;
        address="N/A";
        number=num;
    }
    public void Print()
    {
        System.out.println (name + "      " + address + "      " + number);
    }
    public static void main (String[] args) {
        Employee emp;
        emp=new Employee("Sok Dara", "Takeo", 150);
        emp.Print();
        emp=new Employee();
        emp.Print();
        emp=new Employee("Chan Love", 50);
        emp.Print();
    }
}
```

2. Method Overriding

សំដែកលើករបង្កើតនូវ Method ដែលមានរយៈដូចគ្នា ទាំងនេះនៅក្នុង Super Class និង Sub Class ពេលគឺ Sub Class អាចយក Method ដែលមានក្នុង Super Class មកបន្ថែមលក្ខណៈអាយកថ្មី។

ឧទាហរណ៍

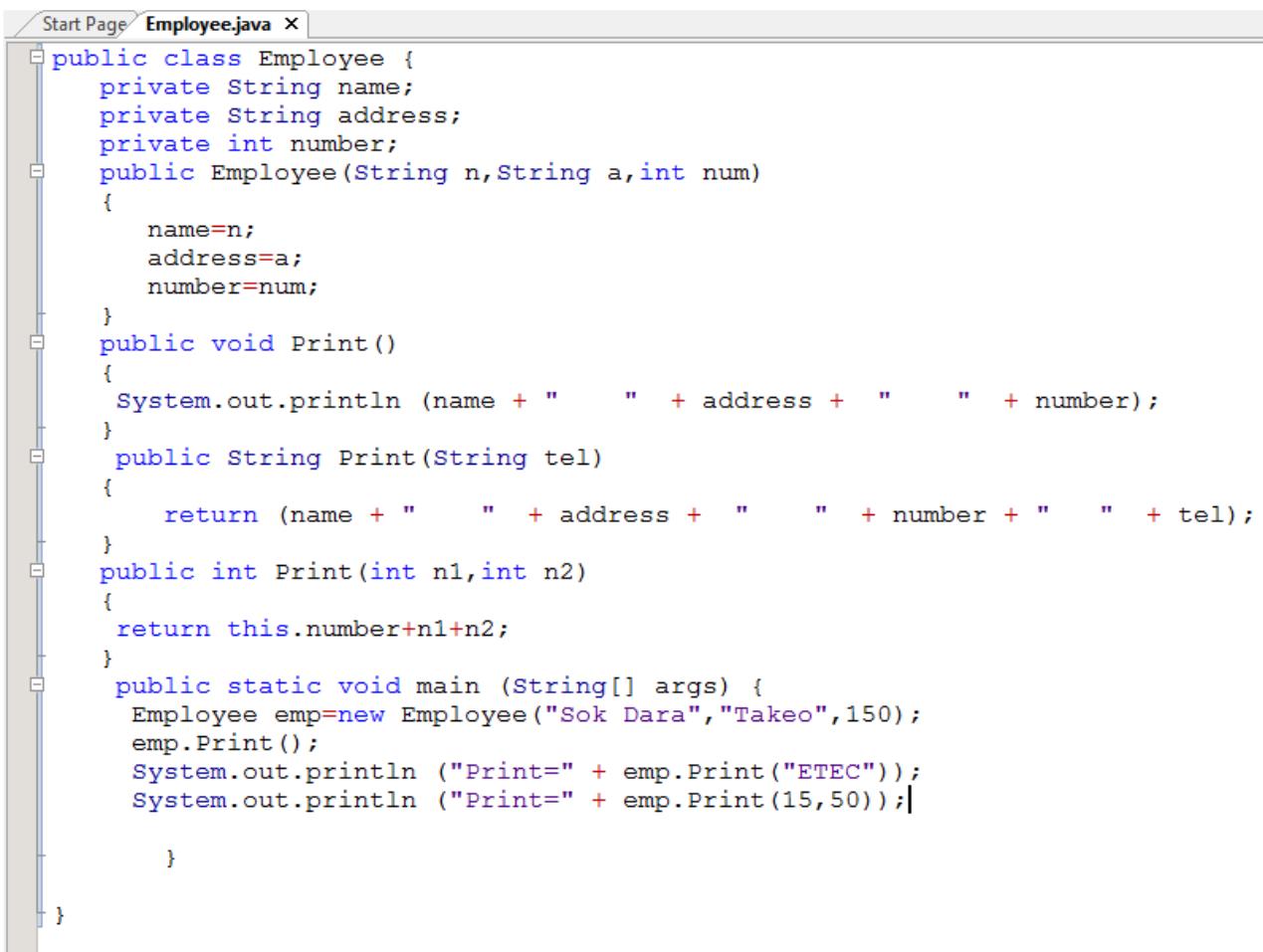
```
class Animal {  
    public void move() {  
        System.out.println("Animals can move");  
    }  
}  
  
class Dog extends Animal {  
    public void move() {  
        super.move(); // invokes the super class method  
        System.out.println("Dogs can walk and run");  
    }  
}  
  
class Fish extends Animal {  
    public void move() {  
        super.move(); // invokes the super class method  
        System.out.println("Fish can travel in water");  
    }  
}  
  
public class TestDog {}  
  
public static void main(String args[]) {  
    Animal b = new Dog(); // Animal reference but Dog object  
    b.move(); // runs the method in Dog class  
    Animal b1 = new Animal();  
    b1.move();  
}
```

នេត្តក្នុងនៃបច្ចេកទេស Method Overriding

- Parameter ត្រូវតែដូចគ្នាអំពី Class Super និង Sub Class
- Return type ត្រូវតែដូចគ្នាអំពី Class Super និង Sub Class
- Method ជាលក្ខណៈ final និង Static មិនអាច overriding បានទេ
- Constructor មិនអាច Overriding បានទេ

Concept ពិនិត្យក្នុងនៃបច្ចេកទេស Polymorphism

- Compile Time or Static polymorphism: សំដែរលើការធ្វើការនៅលើជំនាញកំណត់កាល Compile Code ជាលើកដំបូង។



```
Start Page Employee.java x
public class Employee {
    private String name;
    private String address;
    private int number;
    public Employee(String n, String a, int num)
    {
        name=n;
        address=a;
        number=num;
    }
    public void Print()
    {
        System.out.println (name + " " + address + " " + number);
    }
    public String Print(String tel)
    {
        return (name + " " + address + " " + number + " " + tel);
    }
    public int Print(int n1, int n2)
    {
        return this.number+n1+n2;
    }
    public static void main (String[] args) {
        Employee emp=new Employee("Sok Dara","Takeo",150);
        emp.Print();
        System.out.println ("Print=" + emp.Print("ETEC"));
        System.out.println ("Print=" + emp.Print(15,50));
    }
}
```

Overloading method គឺជាប្រភេទ method ដែលពិពណ៌នាលើការប្រើប្រាស់លក្ខណៈជាអាយក្រាន់តែដើរដីសង្ឃមនឹង Function ឬកម្មយមនលក្ខណៈនៃចំណាំ Parameter និង ប្រភេទទិន្នន័យទុសត្វាមកធ្វើការជាការស្របនៅក្នុងពេលដែលអ្នកបាន Compile ឬតួចតាមការបញ្ជូនឯកសារ

- Runtime Polymorphism: គឺជាប្រភេទ Concept មួយបែបទៀតដែលវាត្រូវការក្នុងជំណាក់កាល Run Code ។ ក្នុងចំណាំនេះគឺត្រូវដើរដីសង្ឃមនឹង Cocept Overring Method មកពន្លាលើពីលក្ខណៈទាំងអស់នេះ៖

```
class Test1{  
    protected int x;  
    protected int y;  
    public Test1(int x,int y){  
        this.x=x;  
        this.y=y;  
    }  
    public void Print()  
    {System.out.println ("X=" + x + " Y=" + y + "\n");  
    }  
}  
class Test2 extends Test1{  
    protected int z;  
    public Test2(int x,int y,int z)  
    {  
        super(x,y);  
        this.z=z;  
    }  
    public void Print()  
    {  
        System.out.println ("X=" + x + " Y=" + y + " Z=" + z);  
    }  
}  
class Test3 extends Test1{  
    protected int z;  
    public Test3(int x,int y,int z)  
    {  
        super(x,y);  
        z=z;  
    }  
    public void Print()  
    {  
        System.out.println ("X=" + x + " Y=" + y + " Z=" + z);  
    }  
}  
class Employee{  
public static void main (String[] args) {  
    Test1 obj;  
    //Calling Method in Super Class  
    obj=new Test1(100,200);  
    obj.Print();  
    //Calling Method in Sub Class  
    obj=new Test2(100,200,300);  
    obj.Print();  
    obj=new Test3(10,20,30);  
    obj.Print();  
}
```

II. ផ្តល់ព័ត៌មានលម្អិតទៅក្នុងរបៀបរាយការណ៍

Abstract Class គឺជាប្រភេទ Class ដែលបានគេត្រូវដោយ Keyword Abstract សេចក្តីថ្លែងជា method Abstract មួយយ៉ាងតិច។

Method Abstract គឺជាប្រភេទ Method ដែលមានតម្លៃយោង ប្រកាសតែត្រូវខ្លួន គឺជាប្រភេទនេះ Class Abstract និង Method Abstract សំរាប់អាយុយ Sub Class អាច Extend ទៅប្រើនិង override លើ Method ដែលមានតម្លៃយោង Abstract Class នេះ។

ឧទាហរណ៍

```
abstract class Bank{  
    abstract int getRateOfInterest();  
}  
  
class SBI extends Bank{  
    int getRateOfInterest(){return 7;}  
}  
  
class PNB extends Bank{  
    int getRateOfInterest(){return 8;}  
}  
  
  
class TestBank{  
    public static void main(String args[]){  
        Bank b;  
        b=new SBI();  
        System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");  
        b=new PNB();  
        System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");  
    }  
}
```

ឯកសារនេះ

Rate of Interest is: 7 %
Rate of Interest is: 8 %

ឧទាហរណ៍ ២

```
//Example of an abstract class that has abstract and non-abstract methods
abstract class Bike{
    Bike(){System.out.println("bike is created");}
    abstract void run();
    void changeGear(){System.out.println("gear changed");}
}

//Creating a Child class which inherits Abstract class
class Honda extends Bike{
    void run(){System.out.println("running safely..");}
}

//Creating a Test class which calls abstract and non-abstract methods
class TestAbstraction2{
    public static void main(String args[]){
        Bike obj = new Honda();
        obj.run();
        obj.changeGear();
    }
}
```

ឯកត្រាគារណ៍

```
bike is created
running safely..
gear changed
```

ផែលាកម្មនូវកម្ពស់

ចូរបង្កើតទីនេះ Class ដូចខាងក្រោម៖

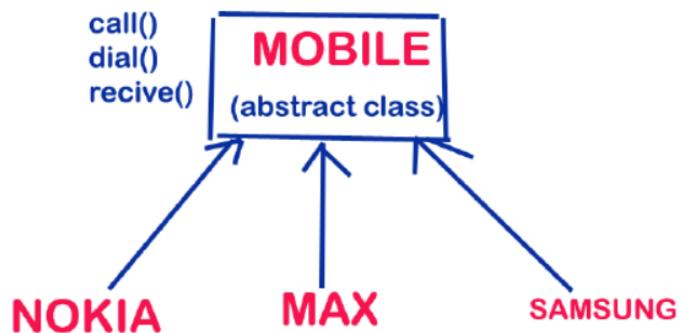


fig: implementing common features to sub classes

Interface ជាឯឹង ?

ឧបាទរណី ទាំង ២ គឺ Interface សំរាប់អោយ Abstract class Implement មកការទៅរាយ

```

interface A{
    void a();
    void b();
    void c();
    void d();
}

abstract class B implements A{
    public void c(){System.out.println("I am c");}
}

class M extends B{
    public void a(){System.out.println("I am a");}
    public void b(){System.out.println("I am b");}
    public void d(){System.out.println("I am d");}
}

class Test5{
    public static void main(String args[]){
        A a=new M();
        a.a();
        a.b();
        a.c();
        a.d();
    }
}
  
```

Output:
I am a
I am b
I am c
I am d

គិតមានទូទៅ Interface មួយដូចខាងក្រោម បូរបញ្ជីតូទៅ Abstract class មួយណ៍ DemoSport ដែល Implement ទៅកាន់ interface Hockey និង Football ហើយបញ្ជីតូទៅ Object ពី Abstract Class យក មកប្រើប្រាស់។

```
// Filename: Sports.java
public interface Sports {
    public void setHomeTeam(String name);
    public void setVisitingTeam(String name);
}

// Filename: Football.java
public interface Football extends Sports {
    public void homeTeamScored(int points);
    public void visitingTeamScored(int points);
    public void endOfQuarter(int quarter);
}

// Filename: Hockey.java
public interface Hockey extends Sports {
    public void homeGoalscored();
    public void visitingGoalscored();
    public void endOfPeriod(int period);
    public void overtimePeriod(int ot);
}
```



“តាមីនាពលជំន្នាករើនបានដឹងថាទុកចាយជាបាន
ប្រាកដមានតំនៃបំផុតក្នុង
បីវត្ថុអ្នករើនបាន...!”

—ក្រុមរាយជីថតនូវ