Generic Programming

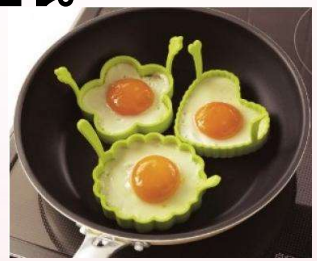
Benjamas Panyangam
Matinee Kiewkanya
Computer Science, CMU

Generic Programming

- การเขียนโปรแกรมแบบเจเนริก (Generic Programming) เป็นวิธีการเขียนโปรแกรม ที่สามารถประกาศตัวแปรที่รองรับข้อมูลได้ทุกชนิด โดยไม่ขึ้นอยู่กับข้อมูลชนิดใดชนิดหนึ่ง
- การเขียนโปรแกรมสั้นและง่าย โปรแกรมมีความยืดหยุ่น
- เช่น หากต้องการเรียงลำดับข้อมูลแต่ละชุด โดยที่ข้อมูลแต่ละชุดมีชนิดข้อมูลแตกต่างกัน การเขียนโปรแกรมแบบเจเนริกจะช่วยให้สามารถเขียนเมท็อดขึ้นมาเพียงเมท็อดเดียวเพื่อทำการเรียงลำดับข้อมูลแบบไม่ระบุชนิดข้อมูล ผ่านเครื่องมือที่เรียกว่า แผ่นแบบ (Template)

แผ่นแบบ (Template)

- **Template หรือแผ่นแบบ** เปรียบเหมือนกับเป็นแบบจำลองในการสร้างสิ่งต่าง ๆ
- ตัวอย่างการใช้งาน Template ในชีวิตประจำวัน
  - แบบจำลองบ้าน
  - แม่พิมพ์ไข่ดาว
  - แม่พิมพ์คุกกี้
  - แม่พิมพ์ทำวุ้น
  - แม่พิมพ์ทำน้ำแข็งในตู้เย็น



Template

การใช้งานแผ่นแบบไว้ 2 รูปแบบคือ

1. แผ่นแบบเมท็อด (Method Template) ใช้สำหรับกำหนดเมท็อดเจเนริก (Generic Method) ที่สามารถทำงานร่วมกับข้อมูลชนิดใดก็ได้
2. แผ่นแบบคลาส (Class Template) ใช้สำหรับกำหนดคลาสเจเนริก (Generic Class) ที่สามารถรับข้อมูลชนิดใดก็ได้

แผ่นแบบเมท็อด (Method Template)

Method Template เป็นการกำหนดต้นแบบของเมท็อด เพื่อให้สามารถใช้เมท็อดเดียวกันกับข้อมูลชนิดต่าง ๆ ผู้เขียนโปรแกรมอาจไม่ต้องเจาะจงชนิดพารามิเตอร์ของเมท็อดประเภทนี้ แต่จะสามารถปรับเปลี่ยนชนิดได้ ขึ้นอยู่กับตอนเรียกใช้งานเมท็อด

ตัวอย่างโปรแกรมที่ 1

โปรแกรมเพื่อหาผลบวกของเลข 2 จำนวน ที่มีชนิดเป็น int ทั้งคู่ หรือ มีชนิดเป็น float ทั้งคู่ โดยใช้ overloaded method

```
public class JavaApp {
    public static int sumNumber(int a, int b)  {
    // sum int
        int result;
        result =a+b;
        return result;
    }

   public static float sumNumber(float a, float b) {
     //sum float
       float result;
       result =a+b;
       return result;
   }
```

```
public static  void main(String[] args){

    int a,b,c;
    a=10;
    b=20;
    c=sumNumber(a,b);
    System.out.print("int c=" + c);

    float x,y,z;
    x=10.2f;
    y=20.3f;
    z=sumNumber(x,y);
    System.out.print("\nfloat z=" + z);
 }  //end main
}  //end class
```

ผลการทำงาน

```
int c=30
float z=30.5
```

ตัวอย่างโปรแกรมที่ 2

โปรแกรมเพื่อหาผลบวกของเลข 2 จำนวน ที่มีชนิดเป็น int ทั้งคู่ หรือ มีชนิดเป็น float ทั้งคู่ โดยใช้ method template

```
public class JavaAppGenericMethod {

 //method template
  public static <TYPE> TYPE sumNumber(TYPE m, TYPE n){

        TYPE result = (TYPE)(Integer)(0);

         if (m.getClass() == Integer.class) {
           result = (TYPE) (Integer) ((Integer) m + (Integer)n) ;
        }
        else if (m.getClass() == Float.class) {
                  result =  (TYPE) (Float) ((Float) m +(Float) n);
        }
         return ((TYPE) result);
}  //end method
```

```
public static  void main(String[] args){

    int a,b,c;
    a=10;
    b=20;
    c=sumNumber(a,b);
    System.out.print("int c=" + c);

    float x,y,z;
    x=10.2f;
    y=20.3f;
    z=sumNumber(x,y);
    System.out.print("\nfloat z=" + z);
  } //end main
} //end class
```

ผลการทำงาน

```
int c=30
float z=30.5
```

Generic Programming

- การตรวจสอบชนิดของ Class สามารถใช้คำสั่ง instanceof แทนได้

  เช่น คำสั่ง if (m.getClass() == Integer.class)

  สามารถเขียนเป็น if (m instanceof Integer)

- จากตัวอย่างข้างต้น ไม่สามารถใช้ได้กับการหาผลรวมของตัวเลขชนิดอื่น ๆ นอกจาก int และ float เนื่องจากไม่ได้ตรวจสอบไว้ เช่น

```
long m,n,r;
m=200;
n=300;
r=sumNumber(m,n);
```

ไม่สามารถเรียกใช้ sumNumberได้ จะ runtime error

ตัวอย่างที่ 3

โปรแกรมเพื่อหาผลบวกของเลข 2 จำนวน ที่มีชนิดเดียวกัน โดยใช้ method template

```
public class JavaAppGenericMethod {

//method template
public static <TYPE extends Number >  double sumNumber(TYPE m, TYPE n)
                {

        double result;
        result = m.doubleValue() + n.doubleValue();
        return result;
  }
```

เมท็อด sumNumber() จะเป็นการบวกเลข 2 ค่าที่มีชนิดเดียวกัน แล้วส่งค่ากลับเป็นชนิด double

```
public static  void main(String[] args){
    double r;

    int a,b;
    a=10;
    b=20;
    r=sumNumber(a,b);
    System.out.print("\nint r=" + (int)r);

    float x,y,z;
    x=10.2f;
    y=20.3f;
    r=sumNumber(x,y);
    System.out.print("\nfloat r=" + (float)r);

    long m,n;
    m=200;
    n=300;
    r=sumNumber(m,n);
    System.out.print("\nlong r=" + (long)r);
  }  //end main
}    //end class
```

ผลการทำงาน

```
int r=30
float r=30.5
long r=500
```

ตัวอย่างโปรแกรมที่ 4

โปรแกรมเพื่อหาผลบวกของเลข 2 จำนวน ที่มีชนิดต่างกัน
โดยใช้ method template ที่มีต้นแบบชนิดข้อมูลหลายชนิด

```
public class JavaAppGenericMethod {

  //method template
  public static <T1 extends Number, T2 extends Number >
            double sumNumber(T1 m, T2 n) {

                Double result;
                result = m.doubleValue() + n.doubleValue();
                return result;
  }
```

```
public static  void main(String[] args){

    double r;
    int a;
    float b;

    a=10;
    b=20.5f;
    r=sumNumber(a,b);
    System.out.print(" r=" + r);

}
```

ผลการทำงาน

r=30.5

สรุปการทำงานของเมท็อด sumNumber() ในโปรแกรมนี้ คือการบวกเลขชนิด int กับชนิด float แล้วส่งค่ากลับเป็นชนิด double

ตัวอย่างโปรแกรมที่ 5

โปรแกรมเพื่อเรียงลำดับแบบฟอง (Bubble Sorting) สำหรับข้อมูลชนิดใดก็ได้ โดยใช้ method template

```
public class JavaAppGenericMethod {
    public static final int N = 5; //constant

    //method template
    public static <TYPE extends Comparable>
                                  void MySort(TYPE data[]) {
        TYPE temp;
        int i,j;

        System.out.print("\nBefore sorting data\n");
        for (i=0; i < N; i++)
                System.out.print(data[i]+"\t");
```

```
for (i=0;i<N; i++){
      for (j=i+1; j<N; j++){
         if(data[i].compareTo(data[j])>0){
            temp=data[i];
            data[i]=data[j];
            data[j]=temp;
         }
      }
   }

   System.out.print("\nResult of sorted data\n");
   for (i=0; i<N; i++)
      System.out.print(data[i]+"\t");


System.out.print("\n**********************\n");
}  //end of method template
```

```
public static  void main(String[] args) {
   Character [] cData = new Character [N];
   Integer [] iData = new Integer [N];
   Float [] fData = new Float[N];
   int i;

   for (i=0; i<N; i++)  {
      cData[i]=(char)((int)(Math.random()*26)+65);
       //random A-Z

      iData[i]= (int)(Math.random()*40)+1;
        //random 1-40
      fData[i]=(float) ((int)(Math.random()*40+1))/3.0f;
      //random 0.33-13.33
     }
 MySort(cData);
 MySort(iData);
 MySort(fData);
}  //end main
}  //end class
```

ผลการทำงาน

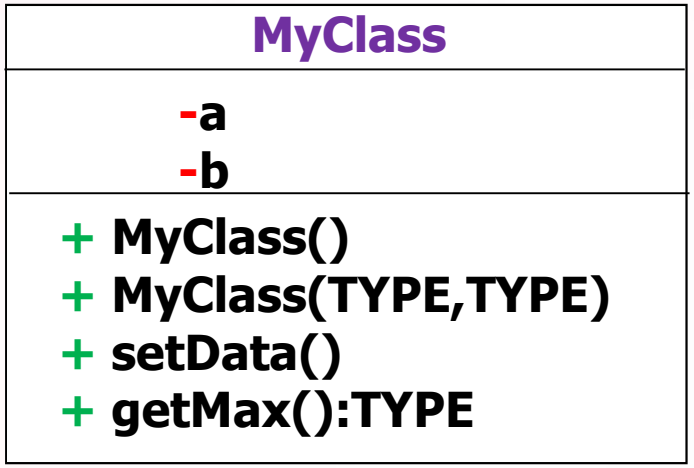
```
Before sorting data
P       G       M       Y       L
Result of sorted data
G       L       M       P       Y
**********************

Before sorting data
28      10      39      26      28
Result of sorted data
10      26      28      28      39
**********************

Before sorting data
5       1.66667         1       8.66667         0.666667
Result of sorted data
0.666667                1       1.66667         5       8.66667
**********************
```

แผ่นแบบคลาส(Class Template)

Class Template เป็นการกำหนดต้นแบบของคลาส โดยที่อาจไม่ต้องเจาะจงชนิดของข้อมูลสมาชิกของคลาส แต่จะสามารถปรับเปลี่ยนชนิดได้ ขึ้นอยู่กับตอนสร้างวัตถุของคลาส ทำให้ข้อมูลภายในคลาสไม่ขึ้นกับชนิดใด ๆ

ตัวอย่างโปรแกรมที่ 6

โปรแกรมเพื่อหาค่าสูงสุดของข้อมูล 2 จำนวน โดยใช้แผ่นแบบคลาส



```
import java.util.Scanner;
//class template
class Myclass <TYPE extends Comparable> {
    private TYPE a;
    private TYPE b;
    Myclass(){} //constructor
    Myclass(TYPE first, TYPE second) {//constructor
        a=first;
        b=second;
     }
```

```
public void setData() {
        Scanner input = new Scanner (System.in);
        System.out.print("Input first data : ");
        a=(TYPE)input.next();
        System.out.print("Input second data : ");
        b=(TYPE)input.next();
}
public TYPE getMax ()   {
     TYPE result = a;
        if (result.compareTo(b)  < 0)
                result = b;
        return result;
 }
} //end of Myclass
```

```
public class JavaAppGenericclass {
    public static void main(String[] args) {
      Myclass <Integer> obj1 = new Myclass(100, 75);
      System.out.println("obj1.getMax = " + obj1.getMax());

      Myclass <Character> obj2  = new Myclass('X','Z');
      System.out.print("\nobj2.getMax = " + obj2.getMax());

      Myclass <Integer> obj3 = new Myclass();
      System.out.print("\n\nInput for obj3\n");
      obj3.setData();
      System.out.print("\nobj3.getMax = " + obj3.getMax());

      Myclass <Character> obj4= new Myclass();
      System.out.print("\n\nInput for obj4\n");
      obj4.setData();
      System.out.print("\nobj4.getMax = " + obj4.getMax());
  }
}
```

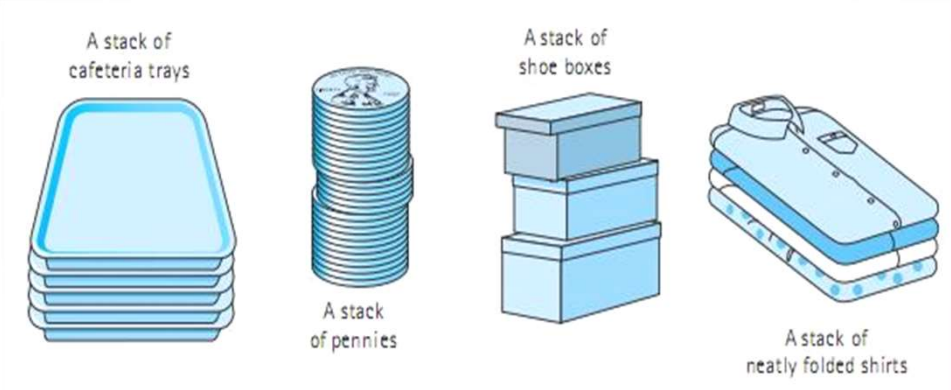
ตัวอย่างผลการทำงาน

```
obj1.getMax = 100
Obj2.getMax = Z

Input for obj3
Input first data : 5
Input second data : 10

Obj3.getMax = 10

Input for obj4
Input first data : x
Input second data : p

Obj4.getmax = x
```

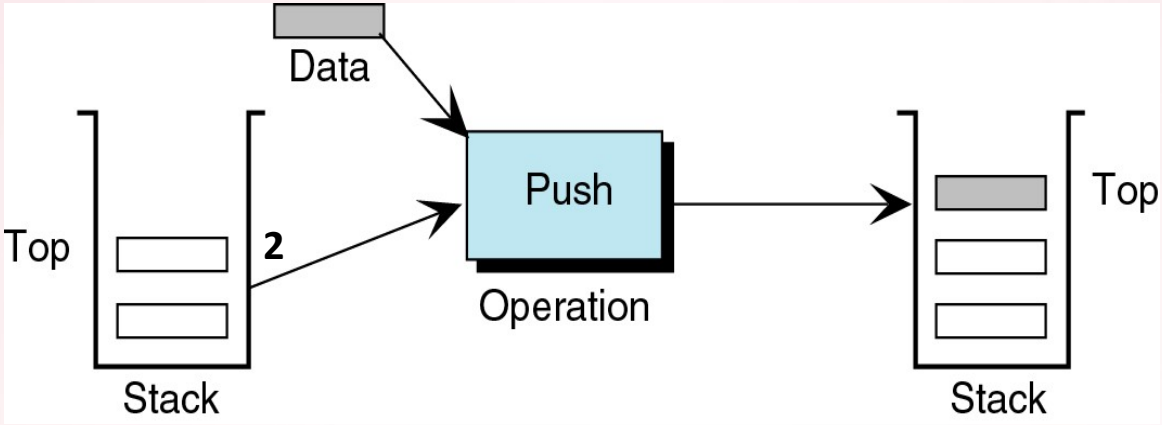
ตัวอย่างโปรแกรมที่ 7

โปรแกรมเพื่อสร้างและดำเนินการกับสแตกโดยใช้แผ่นแบบคลาส

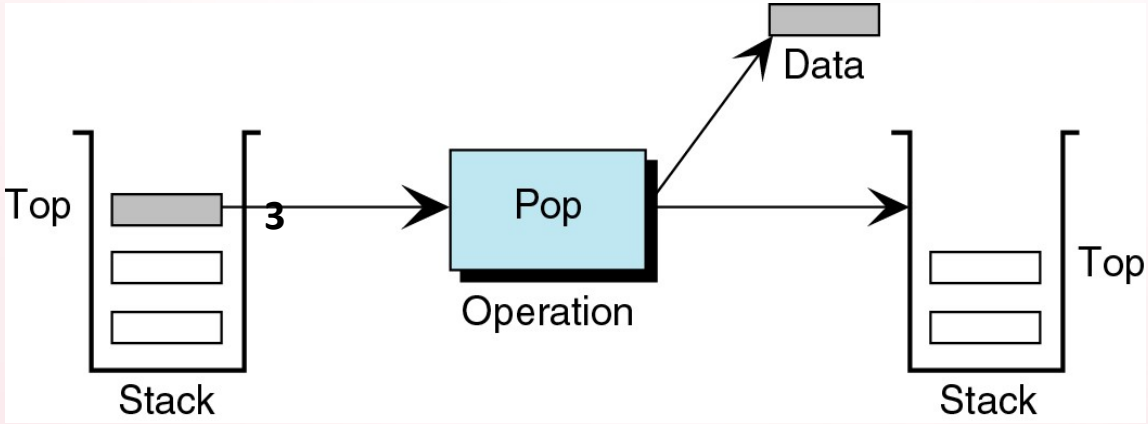
ชนิดข้อมูลแบบกองซ้อน หรือสแตก (Stack) เป็นชนิดข้อมูลแบบนามธรรมที่มีโครงสร้างข้อมูลในลักษณะที่เรียกว่าเข้าก่อนออกทีหลัง (Last In First Out



ตัวอย่างสแตกในชีวิตประจำวัน



การดำเนินการ push



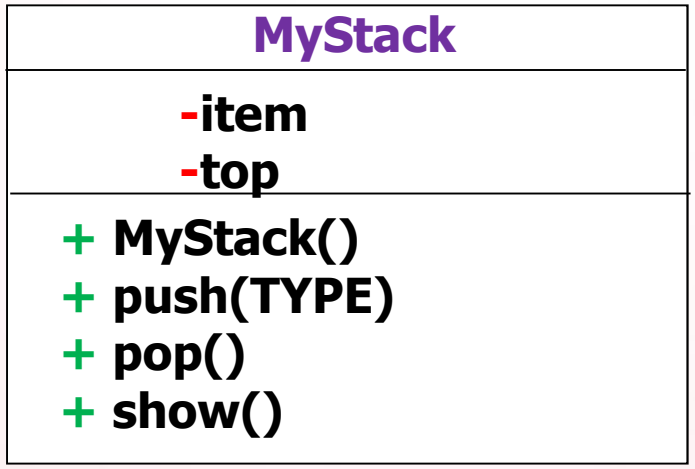
การดำเนินการ pop

ในตัวอย่างนี้จะเป็นการสร้างแผ่นแบบคลาส **MyStack** โดยมีการทำงานตามเมท็อดต่าง ๆ ดังนี้

-**push()** คือการนำข้อมูลเข้าสู่ด้านบนสุดของสแตก

-**pop()** คือการนำข้อมูลที่อยู่ด้านบนสุดออกจากสแตก

-**show()** คือการแสดงข้อมูลทั้งหมดที่อยู่ในสแตก



```
class MyStack <TYPE> { //class template
    private int Max;
    private final Object [] item;
    private int top;

    MyStack(int n) {
          item = new Object[n];
          Max = n;
          top=-1;
    }
    public void push(TYPE v) {
       if (top < Max-1){
           top++;
           item[top]= v;
        }
        else
           System.out.print("\nThe stack is full! cannot
add item "+v);
   }
```

```
public void pop() {
   if(top==-1)
     System.out.print("\nThe stack is empty! cannot get item");
   else  {
     System.out.print("(Got "+ item[top] +")\n");
     top--;
   }
}

 public void show() {
     if(top!=-1) {
          System.out.print("\nThe stack items are\n");
          for(int i=0;i<=top;i++)
             System.out.print( item[i]+"\t");
          System.out.print("\n\n");
      }
      else
          System.out.print("\nNothing to print");
  }
}  //end class MyStack
```

```
public class JavaAppGenericclass {
    public static void main(String[] args) {
        MyStack <Integer> stkI = new MyStack(3);

        stkI.push(8);
        stkI.push(4);
        stkI.push(10);
        stkI.show();

        stkI.pop();
        stkI.pop();
        stkI.pop();
        stkI.pop ();          //Stack empty
        stkI.show();          //Nothing to print

        System.out.print("\n\n**************\n");
```

```
        MyStack<Float> stkF = new MyStack(3);

        stkF.push(8.2f);
        stkF.push(4.1f);
        stkF.push(10.5f);
        stkF.push(2.4f);          //Stack full
        stkF.push(6.3f);          //Stack full
        stkF.show();

        stkF.pop();
        stkF.pop();
        stkF.pop();
        stkF.pop();               //Stack empty
        stkF.show();              //Nothing to print
} //end main
} //end class
```

ผลการทำงาน

```
The stack items are
8       4       10

(Got 10)
(Got 4)
(Got 8)

The stack is empty! cannot get item
Nothing to print

**************
```

ผลการทำงาน(ต่อ)

```
The stack is full! cannot add item 2.4
The stack is full! cannot add item 6.3
The stack items are
8.2    4.1    10.5

(Got 10.5)
(Got 4.1)
(Got 8.2)

The stack is empty! cannot get item
Nothing to print
```