เขียนโปรแกรมภาษา Java

(Phase 3 - หัวข้อเสริม)

ขอบเขตเนื้อหา

- Enums
- Generics
- Collections
- Packages

เครื่องมือพื้นฐาน

- JDK ประกอบด้วย Compiler และ Debugger
- NetBeans IDE เครื่องมือในการพัฒนาโปรแกรม เป็น Editor สำหรับอำนวยความสะดวกในการเขียนโปรแกรมภาษา Java

รู้จักกับ Enum

Enum คือ สิ่งที่ระบุขึ้นเองหรือหมายถึงตัวแปรที่เป็นรูปแบบค่าคงที่ (Constant) โดยมีการตั้งชื่อเฉพาะขึ้นมาเพื่อเป็นตัวแทนของกลุ่มข้อมูล (นิยามชื่อชนิดข้อมูลเอง)

คุณสมบัติของ Enum

- Enum ใช้เป็นตัวแทนของกลุ่มข้อมูลที่เป็นค่าคงที่
- ตัวแปรชนิด Enum เก็บได้เฉพาะค่าที่ประกาศใน Enum เท่านั้น
- ไม่สามารถนำตัวแปร Enum มาใช้ในการคำนวณได้
- สามารถสร้าง Method และ Attribute ใน Enum ได้

การสร้าง Enum

```
enum ชื่อEnum {
        value1,
        value2,
        .....
}
```

ตัวอย่าง Enum

- enum เพศ {ชาย , หญิง}
- enum ฤดู {ร้อน , ฝน , หนาว}
- enum เกรด {A , B , C , D, F}
- enum วัน {อาทิตย์ , จันทร์ , อังคาร , ... , เสาร์}
- enum ระดับ {ง่าย , ปานกลาง , ยาก}
- enum สีสัญญานไฟจราจร {แดง , เหลือง , เขียว}

Generic คืออะไร

เป็นกระบวนการจัดการประเภทข้อมูลที่ระบุอยู่ภายในคลาสและเมธอดให้มีความยืดหยุ่นตามการเรียกใช้งาน โดยรูปแบบการระบุประเภทข้อมูลนั้นจะเขียนในพื้นที่ <>

Generic Classes

หรือเรียกอีกชื่อคือ Parameterized Class หมายถึงคลาสที่สามารถมีพารามิเตอร์ได้

โดยพารามิเตอร์ของ Generic Class จะถูกเรียกว่า “Type Parameter” สำหรับจัดการเกี่ยวกับประเภทข้อมูลภายในคลาส

Wrapper Classes

Primitive Data Type	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long

Wrapper Classes

Primitive Data Type	Wrapper Class
float	Float
double	Double
char	Character
boolean	Boolean

Generic Classes

โครงสร้างคำสั่ง

```
class ClassName<T>{


}
```

Generic Classes

โครงสร้างคำสั่ง

type parameter

```
class ClassName<T>{


}
```

Generic Classes

โครงสร้างคำสั่ง

ให้ T เป็นตัวแทนของประเภทข้อมูล
หรือคลาสที่สนใจ

```
class ClassName<T>{


}
```

<T> คืออะไร , ทำไมต้องใช้ T

เราสามารถใช้ตัวอักษรอื่นแทน T ได้ เช่น X , Y , Z เนื่องจากตัวอักษรดังกล่าวเปรียบเสมือนกับการนิยามตัวแปรขึ้นมาใช้งาน เพื่อเป็นตัวแทนของประเภทข้อมูลหรือคลาสที่เราสนใจเท่านั้น เช่น

ตัวอย่างตัวอักษรที่นิยมใช้งาน

T - Type (ชนิดข้อมูล)

E - Element (สมาชิก)

K - Key (คีย์)

V - Value (ข้อมูล)

<T> คืออะไร , ทำไมต้องใช้ T

เราสามารถใช้ตัวอักษรอื่นแทน T ได้ เช่น X , Y , Z เนื่องจากตัวอักษรดังกล่าวเปรียบเสมือนกับการนิยามตัวแปรขึ้นมาใช้งาน เพื่อเป็นตัวแทนของประเภทข้อมูลหรือคลาสที่เราสนใจเท่านั้น เช่น

ตัวอย่างตัวอักษรที่นิยมใช้งาน

- T – Type (ชนิดข้อมูล)
- E – Element (สมาชิก)
- K – Key (คีย์)
- V – Value (ข้อมูล)

Generic Classes

การเรียกใช้งาน

```
ClassName<type1> objName = new ClassName<>(param);

ClassName<type2> objName = new ClassName<>(param);

ClassName<type3> objName = new ClassName<>(param);

ClassName<type4> objName = new ClassName<>(param);
```

Generic Classes

การเรียกใช้งาน

```
ClassName<type1> objName = new ClassName<>(param);
ClassName<type2> objName = new ClassName<>(param);
ClassName<type3> objName = new ClassName<>(param);
ClassName<type4> objName = new ClassName<>(param);
```

Bounded Type Parameter

คือ การกำหนดข้อบังคับหรือขอบเขตในการจัดการประเภทข้อมูลที่อยู่ใน Type Parameter

โครงสร้างคำสั่ง

```
class ClassName<T extends type>{
}
```

Multiple Type Parameter

โครงสร้างคำสั่ง

```
class ClassName<T , U>{


}
```

Multiple Type Parameter

โครงสร้างคำสั่ง

ให้ T เป็นตัวแทนของประเภทข้อมูลตัวที่ 1
และ U เป็นตัวแทนของประเภทข้อมูลตัวที่ 2

```
class ClassName<T , U>{


}
```

Generic Method

คือ การสร้างเมธอดที่สามารถจัดการข้อมูลต่างกันได้

โครงสร้างคำสั่ง (void method)

```
<T> void methodName(T parameter){
        //คำสั่งต่างๆ
}
```

Generic Method

คือ การสร้างเมธอดที่สามารถจัดการข้อมูลต่างกันได้

โครงสร้างคำสั่ง (return method)

```
<T> T methodName(T parameter){
        //คำสั่งต่างๆ
}
```

Generic Method

คือ การสร้างเมธอดที่สามารถจัดการข้อมูลต่างกันได้

โครงสร้างคำสั่ง

ระบุว่าเป็น Generic โดยให้ T เป็นตัวแทนของประเภทข้อมูลที่สนใจ

```
<T> void methodName(T parameter){
        //คำสั่งต่างๆ
}
```

คอลเล็กชัน (Collection)

- เป็นตัวจัดการกลุ่มข้อมูลที่ใช้เก็บข้อมูลหลายๆค่าเอาไว้ด้วยกัน ทำให้เราจัดการข้อมูลได้ง่ายและสะดวกมากขึ้นอีก ทั้งยังมีความยืดหยุ่น สามารถเพิ่มและลดขนาดได้เองอัตโนมัติตามข้อมูลที่มีอยู่

- การใช้งาน Collection จะเรียกใช้งานผ่าน package java.util

ตัวอย่างคอลเล็กชัน (Collection)

- ArrayList - จัดการข้อมูลแบบ Array ในรูปแบบของ Dynamic Size
- HashSet - จัดการข้อมูลไม่ซ้ำกัน
- HashMap - จัดการความสัมพันธ์ของข้อมูล

การใช้งานคอลเล็กชัน (Collection)

- import java.util.ชื่อคลาส;
- import java.util.*;

ArrayList

คลาส ArrayList เป็นหนึ่งในกลุ่ม Collection ที่ใช้เก็บข้อมูลเหมือน Array

แต่ ArrayList นั้นสามารถย่อและขยายขนาดการเก็บข้อมูลได้อัตโนมัติ

ข้อจำกัดของ Array

- การสร้าง Array นั้นจะต้องกำหนดขนาดของ Array เสมอ เมื่อกำหนดแล้วจะไม่สามารถเปลี่ยนแปลงได้ (Fixed Size) และไม่สามารถเก็บข้อมูลได้เกินกว่าขนาดที่กำหนด

- การสร้าง Array ต้องรู้ว่าจะเก็บข้อมูลอะไรบ้าง จึงจะสามารถกำหนดขนาด Array ได้อย่างถูกต้อง ซึ่งเป็นเรื่องที่ยากมาก

ใช้ ArrayList แทน Array

- ArrayList ทำงานคล้ายกับ Array มีการเก็บข้อมูลและอ้างอิงตำแหน่งของข้อมูลผ่านหมายเลขกำกับ (Index)

- มีความยืดหยุ่นในการเก็บข้อมูลสามารถเพิ่มหรือลดขนาดได้ตามความต้องการ (Dynamic Size)

Array VS ArrayList

Array	ArrayList
ขนาดคงที่ (Fixed Size)	ขนาดมีความยืดหยุ่น (Dynamic Size)
ต้องระบุขนาดล่วงหน้า	ไม่ต้องระบุขนาดล่วงหน้า
ใช้เก็บข้อมูลที่เป็นชนิดพื้นฐานหรือวัตถุ • int[] • String[]	ใช้เก็บวัตถุ (คลาส Object) • ArrayList<Integer> • ArrayList<String>

การสร้าง ArrayList

- import java.utill.ArrayList;
- ArrayList <class_name> ชื่อตัวแปร = new ArrayList<>();

เมธอดจัดการ ArrayList

ชื่อเมธอด	คำอธิบาย
add(element)	เพิ่มสมาชิกใหม่ 1 รายการ
add(index,element)	เพิ่มสมาชิกใหม่ 1 รายการไปยัง index ที่กำหนด
addAll([element])	เพิ่มสมาชิกใหม่หลายรายการ
addAll(index ,[elementt])	เพิ่มสมาชิกใหม่หลายรายการไปยัง index ที่กำหนด

เมธอดจัดการ ArrayList

ชื่อเมธอด	คำอธิบาย
size()	จำนวนสมาชิกทั้งหมดใน ArrayList
get(index)	ดึงข้อมูลสมาชิกใน ArrayList จาก index ที่กำหนด
set(index,element)	เปลี่ยนแปลงข้อมูลสมาชิกใน ArrayList จาก index ที่กำหนด
contains(element)	ตรวจสอบว่ามีสมาชิกใน ArrayList หรือไม่

เมธอดจัดการ ArrayList

ชื่อเมธอด	คำอธิบาย
clear()	ลบสมาชิกทั้งหมดออกจาก ArrayList
remove(element)	ลบสมาชิกที่ระบุออกจาก ArrayList
remove(index)	ลบสมาชิกตาม index ที่ระบุออกจาก ArrayList
indexOf(element)	ตรวจสอบลำดับสมาชิกใน ArrayList

HashSet

มีลักษณะการทำงานคล้ายกับ ArrayList โดยข้อมูลหรือสมาชิกที่เก็บใน HashSet นั้นต้องมี“ค่าไม่ซ้ำกัน” แต่สมาชิกใน HashSet นั้นจะไม่มีการเรียงลำดับที่ถูกเพิ่มเข้ามา

การสร้าง HashSet

- import java.utill.HashSet;
- HashSet <class_name> ชื่อตัวแปร = new HashSet<>();

เมธอดจัดการ HashSet

ชื่อเมธอด	คำอธิบาย
add(element)	เพิ่มสมาชิกใหม่ 1 รายการ
size()	จำนวนสมาชิกทั้งหมดใน HashSet
contains(element)	ตรวจสอบว่ามีสมาชิกใน HashSet หรือไม่

เมธอดจัดการ HashSet

ชื่อเมธอด	คำอธิบาย
clear()	ลบสมาชิกทั้งหมดออกจาก HashSet
remove(element)	ลบสมาชิกที่ระบุออกจาก HashSet

LinkedHashSet , TreeSet

- LinkedHashSet มีลักษณะคล้ายกับ HashSet แต่สมาชิกจะเรียงลำดับที่ถูกเพิ่มเข้ามา
- TreeSet มีลักษณะคล้ายกับ HashSet แต่สมาชิกจะเรียงลำดับจากน้อยไปมาก , ตามลำดับตัวอักษร หรือลำดับอื่นๆตามที่ผู้ใช้กำหนด

HashMap

เป็นรูปแบบการเก็บความสัมพันธ์ของข้อมูลในลักษณะของ key และ value

กล่าวคือ จะใช้ key เป็น index ในการเข้าถึงข้อมูล (value) แต่ละตัว ซึ่งค่า key นั้นต้องไม่ซ้ำกัน

ArrayList VS HashMap

หมายเลขกำกับ (Index)	ชื่อประเทศ
0	ประเทศไทย
1	ประเทศญี่ปุ่น
2	ประเทศจีน

ArrayList VS HashMap

รหัสประเทศ (คีย์)	ชื่อประเทศ
TH	ประเทศไทย
JP	ประเทศญี่ปุ่น
CN	ประเทศจีน

ตัวอย่างการประยุกต์ HashMap

ตัวย่อ(คีย์)	สกุลเงิน
THB	บาท
EUR	ยูโร
GBP	ปอนด์

ตัวอย่างการประยุกต์ HashMap

สถานะ (คีย์)	ความหมาย
True	ตกลง
False	ยกเลิก

การสร้าง HashMap

HashMap<key_type , value_type> ชื่อตัวแปร = new HashMap<>();

- key_type คือ ประเภทข้อมูลของคีย์
- value_type คือ ประเภทข้อมูลของ value

เมธอดจัดการ HashMap

ชื่อเมธอด	คำอธิบาย
put(key,element)	เพิ่มสมาชิกใหม่
get(key)	ดึงสมาชิกจากคีย์ที่กำหนด
size()	จำนวนสมาชิกใน HashMap

เมธอดจัดการ HashMap

ชื่อเมธอด	คำอธิบาย
clear()	ลบสมาชิกทั้งหมดออกจาก HashMap
remove(key)	ลบสมาชิกตามคีย์ที่ระบุออกจาก HashMap
containsKey(key)	ตรวจสอบว่ามีคีย์ใน HashMap หรือไม่
containsValue(value)	ตรวจสอบว่ามีค่าข้อมูลใน HashMap หรือไม่

แพ็กเกจ (Package)

คือ สิ่งที่ช่วยให้ผู้พัฒนาโปรแกรมสามารถจัดการ Class และ Interface ออกเป็นกลุ่มการทำงานแต่ละส่วนได้ ด้วยวิธีการแยกไฟล์ออกเป็นหมวดหมู่ต่างๆ ตามรูปแบบการทำงาน

ส่งผลให้โค้ดมีความเป็นระเบียบและนำกลับมาใช้งานใหม่ได้ง่ายและสะดวกมากยิ่งขึ้น

แพ็กเกจ (Package)

1. การนำคลาสมาใส่แพ็กเกจ
2. แพ็กเกจย่อย
3. การอ้างอิงคลาสในแพ็กเกจ
4. การ Import คลาสจากแพ็กเกจอื่น

แพ็กเกจ (Package)

- การนำคลาสมาใส่แพ็กเกจ

  package ชื่อแพ็กเกจ;

- แพ็กเกจย่อย

  package ชื่อแพ็กเกจหลัก.ชื่อแพ็กเกจย่อย;

- การอ้างอิงคลาสในแพ็กเกจ

  ชื่อแพ็กเกจ.ชื่อคลาส;

การ Import คลาสจากแพ็กเกจอื่น

- import package_name.class_name;

  หมายถึง import เฉพาะคลาสที่สนใจใน package

- import package_name.*;

  หมายถึง import ทุกคลาสที่อยู่ใน package

ช่องทางการสนับสนุน

🕵️ ช่องยูทูป :https://www.youtube.com/c/KongRuksiamOfficial

🎓 คอร์สเรียน : https://www.udemy.com/user/kong-ruksiam/

🛒 ซื้อของผ่าน Shopee : https://shope.ee/3plB9kVnPd

🎯 แฟนเพจ : https://www.facebook.com/KongRuksiamTutorial/