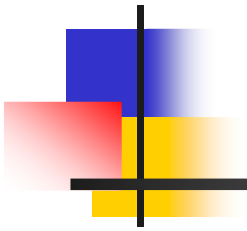


CPE111

Programming with Data Structures (Java)

การสร้าง Java Application

1. ใช้โปรแกรม **Text Editor** เขียนคำสั่งที่ต้องการในรูปแบบของภาษาจาวา โดยมักใช้ชื่อคลาสเป็นชื่อไฟล์ เก็บในไฟล์ที่มีนามสกุลเป็น **".java"** เช่น **"hello.java"** ฯลฯ โดยจะต้องมีคลาสหลักซึ่งมีฟังกชันชื่อ **main()** ปรากฏอยู่
2. แปล(**Compile**) โปรแกรมที่เขียนขึ้น โดยใช้ **"javac.exe"**ขั้นตอนนี้จะได้ผลลัพธ์เป็นชื่อ **class** นามสกุล **".class"** เช่น **javac hello.java** จะได้ไฟล์ชื่อ **"hello.class"**
   - ในกรณีมีหลายๆไฟล์ หรือ หลายๆคลาส สามารถใช้ **wildcard** * แทนได้เช่น **"javac *.java"**
   - โดยปกตินิยมเขียน **1** คลาส ไว้ใน **1** ไฟล์ กรณีมีหลายคลาสใน **1** ไฟล์ **javac** จะสร้างไฟล์ **".class"** เพิ่มให้ตามจำนวนคลาสที่อยู่ในไฟล์
   - ในกรณีที่โปรแกรมจาวามีการอ้างถึงคลาสอื่นๆ ที่ไม่ได้อยู่ใน **current directory**สามารถกำหนด **CLASSPATH** เพื่อให้ **javac.exe** หาคลาสที่เกี่ยวข้องกับโปรแกรมที่เขียนขึ้นได้เอง (กรณีต้องการใช้คลาสที่เคยเขียนไว้แล้วร่วมกันกับโปรแกรมอื่น ไม่ต้องเขียนใหม่)
3. เมื่อต้องการรันโปรแกรมให้เรียกผ่าน **"java.exe"** โดยจะต้องรันโปรแกรมของคลาสที่มี **main()** ปรากฏอยู่ในคลาสนั้น เช่น **"java hello" (รันโปรแกรม hello.class)**

Eclipse IDE for Java Developers

- เริ่มต้นเขียนโปรแกรมครั้งแรก เปิดโปรแกรม eclipse
  1. สร้าง project -> File -> new -> java project
     - ตั้งชื่อโปรเจคเช่น Test Lab ซึ่งจะกลายเป็นชื่อ folder ใน workspace ที่กำหนดไว้ -> Finish
  2. ถ้าไม่ระบุจะเป็น default package
     - ถ้าต้องการสร้าง package ใหม่ -> File -> new -> Package
  3. สร้าง class หลัก -> File -> new -> Class
     - ตั้งชื่อ class หลักเช่น Lab0 ซึ่งจะกลายเป็น ชื่อไฟล์ Lab0.java อยู่ใน /src
     - class หลักนี้ถ้าติ๊ก public static void main(String[] args) eclipse จะสร้าง main เริ่มต้นให้ (หรือจะเขียนเองก็ได้)
     - เพิ่มส่วน import ที่หัวโปรแกรมตามที่ต้องการ เช่น import java.util.*;
     - สร้าง methods เพิ่มเติมใน class หลัก
  4. ถ้าต้องการสร้าง class เพิ่มเติมให้ทำเช่นเดิม(2) แต่ไม่ต้องเลือกให้สร้าง main
- เข้าโปรแกรมในครั้งต่อไป ให้เข้าไปเลือกไฟล์ ในโปรเจคที่ต้องการได้เลย
- กรณีเปลี่ยนเครื่องทำงาน
  - Copy folder project ที่ต้องการ ไปไว้ใน workspace ของเครื่องใหม่
  - ไปที่ File -> import -> General->Existing Projects into Workspace
  - Select root directory: โดย Browse.. ไปยัง Folder ที่ Copy ไว้

โปรแกรมภาษาจาวา

- ภาษาจาวามีการทำงานคล้ายกับ interpreter แต่จะมีการคอมไพล์ source code (extension.java) ให้กลายเป็น (extension.class) ซึ่งชุดคำสั่งของ Java Virtual Machine(JVM) ก่อน เวลารันโปรแกรมเรียกใช้ก็จะเรียก JVM เพื่อประมวลผลคำสั่งของ JVM (extension.class) ให้กลายเป็นคำสั่งประมวลของคอมพิวเตอร์เครื่องนั้น อีกครั้งหนึ่ง
- ในการเขียนโปรแกรมด้วยภาษาเชิงวัตถุ(Object-Oriented Programming) จะเป็นการกำหนดโครงสร้าง คือคุณสมบัติต่างๆของวัตถุ เช่น ตัวแปร และฟังก์ชั่นที่ใช้ประมวลผล รวมเรียกว่าคลาส (Class) แล้วใช้คำสั่ง new สร้างวัตถุตัวอย่าง (instance) ขึ้นมาเพื่อใช้งาน (เทียบกับแนวคิดภาษาซี class จะคล้ายกับการรวมเอาตัวแปร structure และฟังก์ชันเข้าด้วยกัน ส่วน instance จะคล้ายกับตัวแปรพอยต์เตอร์ ที่มีชนิดข้อมูลเป็นคลาสนั้น ซึ่งการใช้งานจะต้องมีการสร้างเนื้อที่ด้วยคำสั่ง malloc ขึ้นมาก่อน จึงจะใช้งานได้)
- จาวามีการทำงานแบบ dynamic allocation ซึ่งตัววัตถุ(ข้อมูลตัวแปรและฟังก์ชันต่างๆ) จะถูกสั่งให้สร้างขึ้นระหว่างการรันโปรแกรม (ไม่ได้สร้างเตรียมไว้ล่วงหน้าเหมือนกับภาษาซี) และจะถูกลบไปเองเมื่อเลิกใช้งาน(life time) ดังนั้นเมื่อต้องการใช้ตัวแปรหรือคำสั่ง(Method) ของสิ่งใด จะต้องมีการสร้างตัวอย่าง(instance) ที่เกี่ยวข้องกับสิ่งนั้นขึ้นมาก่อน จึงจะทำงานกับตัวแปร หรือเรียกใช้คำสั่งที่เกี่ยวข้องกับสิ่งนั้นได้ โดยเรียกใช้ผ่านทาง instance ที่สร้างขึ้น
- ผู้ใช้สามารถกำหนดให้บางเมท็อดให้เป็นแบบ static เพื่อให้มีการสร้างเตรียมไว้ล่วงหน้า ทำให้เรียกใช้โดยตรงไม่ต้องเรียกผ่าน instance ก็ได้ กรณีที่มีหลายคลาส สามารถเรียกใช้เมท็อดร่วมกันระหว่างคลาสต่างๆ ได้

Public Class & Filename

- การเขียนภาษาจาวา ต้องเริ่มจากการสร้างคลาสหลักของโปรแกรม และ สร้าง method ชื่อ public static void main (String [] args) เพื่อใช้สำหรับรันโปรแกรมที่จะเขียนก่อน โดยตั้งชื่อไฟล์ของโปรแกรม เป็นชื่อเดียวกับชื่อคลาสหลัก

```
public class Hello
{
    public static void main (String [] args)
    {
      System.out.println("Hello World");
    }
}
```

สร้างคลาสหลักชื่อ Hello ตั้งชื่อไฟล์เป็น "Hello.java"

มี method ชื่อ main สำหรับรันโปรแกรม

- ใน 1 ไฟล์ จะมีคลาสที่เป็น public ได้เพียง 1 คลาสเท่านั้น และให้ตั้งชื่อไฟล์ให้ตรงกับคลาสที่เป็น public (สามารถมีหลายคลาสได้ แต่ต้องมีเพียง 1 public)
- กรณีที่โปรแกรมมีหลายไฟล์ หรือมีหลายคลาส ทำงานร่วมกันจะต้องมีคลาสหลักหนึ่งคลาสที่ใช้ RUN ซึ่งคลาสนั้นจะต้องมี public static void main(String args[]) { } เป็นองค์ประกอบ ส่วนคลาสอื่นๆที่เรียกใช้ร่วมกัน จะต้องไม่มีส่วนนี้
- Java นิยมตั้งชื่อคลาสขึ้นต้นด้วยอักษรตัวใหญ่
- การเรียกใช้คำสั่งต่างๆ(Methods) ของ java จะต้องสร้าง Object (หรือเรียกว่า instance) ด้วยคำสั่ง new ขึ้นมาก่อน แล้วจึงเรียกใช้ฟังค์ชันของคลาสที่ต้องการ(Methods) ผ่านทาง instance ตัวนั้น ยกเว้นเมท็อดที่ถูกระบุให้เป็น static สามารถเรียกใช้ได้โดยไม่จำเป็นต้องมี instance

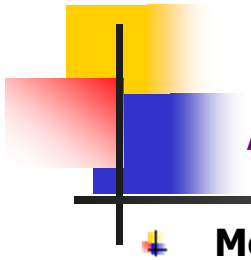
Classes

- **Class คือคุณสมบัติของ Object ซึ่งจะประกอบด้วย**
  - **attribute คือตัวแปร หรือค่าคงที่ ที่ใช้ในคลาส เช่น int i, double x, String str**
  - **Method คือฟังก์ชันที่เตรียมไว้ใช้จัดการกับตัวแปรของคลาสนั้น เช่น การใส่ค่า การหาคำตอบ การเปรียบเทียบเพื่อเรียงลำดับ การแสดงผล ฯลฯ**

การสร้างคลาส ให้ระบุคำว่า class ตามด้วยชื่อคลาสที่ตั้งขึ้น

```
[modifier] class  className {  // modifier  public, private, protect
        [class member]  //  Attributes
[modifier] return_type  methodName ([arguments]) {  // modifier  static, public
        [method_body]
    }
}
```

  - **Public Class เป็นคลาสที่ถูก Save ในไฟล์ในชื่อเดียวกับชื่อคลาส สามารถถูกคลาสอื่นที่สร้างขึ้นมาเรียกใช้ได้**
  - **ใน 1 ไฟล์จะมีคลาสที่เป็น Public ได้เพียง 1 คลาสเท่านั้น คือคลาสที่ตั้งชื่อให้ตรงกับชื่อไฟล์ ถ้าต้องการเพิ่มคลาสอื่นเก็บในไฟล์เดียวกัน คลาสอื่นจะต้องไม่เป็น Public และคลาสนั้นจะถูกมองเห็นได้เฉพาะในไฟล์นั้นเท่านั้น**
- **ในคลาสที่ใช้รันโปรแกรม จะเป็น public class โดยตั้งชื่อไฟล์ให้ตรงกับชื่อคลาสนั้น และจะต้องสร้าง Method ชื่อ public static void main(String[] args) {....} เพื่อใช้รันโปรแกรม**



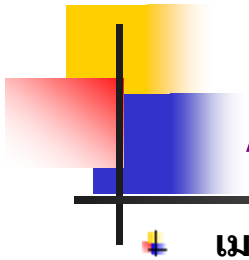
Modifiers

- **Modifier คือคำที่ใช้ระบุถึงการเข้าถึง และใช้งาน ตัวแปร เมท็อด หรือคลาส ใช้เพื่อระบุสิทธิ์ในการเข้าถึงข้อมูลต่างๆที่เราสร้างขึ้น (Encapsulation)**
- **Access Control Modifiers ใช้ควบคุมระดับการเข้าถึงคลาส เมท็อด หรือ ตัวแปร**
  - **public มีขอบเขตการเรียกใช้ได้ทุกระดับ สามารถเรียกใช้ได้จากทุกคลาส**
  - **private มีขอบเขตการเรียกใช้ เฉพาะภายในคลาสเดียวกันเท่านั้น**
  - **protected มีขอบเขตการเรียกใช้ เฉพาะคลาสที่อยู่ภายในแพ็คเกจเดียวกัน หรือ คลาสแม่ที่สืบทอดมาเท่านั้น (Inheritance)**
  - **default มีขอบเขตการเรียกใช้ เฉพาะคลาสที่อยู่ภายในแพ็คเกจเดียวกัน**
- **Non Access Control Modifiers ใช้ควบคุมคุณสมบัติอื่นๆ**
  - **static ใช้ระบุเพื่อให้สร้าง methods และ attributes ไว้ล่วงหน้า**
    - **ใช้ static นำหน้า method ทำให้โปรแกรมสามารถมองเห็น และเรียกใช้ method นั้นได้ตลอดเวลาโดยไม่ต้องสร้าง instance ขึ้นมาก่อน**
      - **ในคลาสเดียวกัน สามารถเรียกชื่อ method ได้โดยตรง**
      - **ต่างคลาส สามารถเรียกใช้ stactic method ผ่านชื่อคลาสได้** //classname.method
    - **static method จะเรียกใช้ method อื่นที่ไม่ระบุ static นำหน้าโดยตรงไม่ได้ ต้องเรียกผ่าน instance ที่สร้างขึ้นใหม่**
    - **ถ้าใช้ static นำหน้าตัวแปร จะทำหน้าที่คล้าย global variable ทำให้เรียกใช้ตัวแปร (memory) ตัวเดียวกัน**

Methods

- **ภาษาเชิงวัตถุจะเรียกโปรแกรมย่อย ว่าเมท็อด(method) (ภาษาซีเรียก ฟังก์ชัน) ถูกสร้างให้เป็นคุณสมบัติหนึ่งของวัตถุ อยู่ภายในคลาส**
  - **เมท็อดถูกสร้างอยู่ภายในคลาสโดยไม่จำเป็นต้องมีลำดับการเห็นก่อนหลัง**
  - **การเรียกใช้เมท็อด จะต้องสร้าง instance ขึ้นมาก่อนแล้วจึงเรียกใช้ผ่าน instance ตัวนั้น (ยกเว้นจะระบุ static)**
  - **เมท็อดที่สร้างขึ้น สามารถอ้างหรือถูกอ้างจากเมท็อดอื่นที่อยู่ภายในคลาสเดียวกัน หรือต่างคลาสกันได้ ขึ้นอยู่กับการระบุ modifier**
    - **เมท็อดที่ไม่ระบุ static สามารถเรียกใช้เมท็อดใดๆก็ได้ที่อยู่ในคลาสเดียวกัน**
    - **ในคลาสเดียวกันเมท็อดที่ระบุ static ไว้ สามารถถูกเรียกโดยเมท็อดอื่นได้**
    - **ในคลาสเดียวกันเมท็อด static จะเรียกเมท็อดที่ไม่ระบุ static มาใช้โดยตรงไม่ได้ ต้องเรียกผ่าน instance**
    - **การเรียกใช้เมท็อดของคลาสอื่น ถ้าเมท็อดที่ต้องการเรียกระบุ static ไว้ ให้อ้างถึงคลาสก่อนแล้วตามด้วย.ชื่อเมท็อด เช่น Classname.methodName() แต่ถ้าไม่ระบุ static ต้องเรียกผ่าน instance เท่านั้น**

	เรียก method ในคลาสเดียวกัน โดยตรง			เรียก method ที่อยู่นอกคลาส	
modifier Method	เรียก static โดยตรง	เรียก non static	เรียกผ่าน instance	static เรียกผ่าน dlass	non staic เรียกผ่าน instance
static	**yes**	**no**	**yes**	**class.method**	**instance.method**
ไม่ระบุ	**yes**	**yes**	**yes**	**class.method**	**instance.method**



Methods

- เมท็อด ทั่วไปที่มักจะสร้างพร้อมกับคลาส
  - การใส่ข้อมูล (constructor สำหรับใส่ข้อมูล 1 ชุด ลงใน instance ที่สร้างขึ้น)
  - การเปรียบเทียบข้อมูล (compareTo กรณีที่ต้องการเรียกใช้การเรียงลำดับที่จาวามีให้)
  - การแสดงข้อมูล (toString แสดงข้อมูลของตัวแปร)
- การสร้างเมท็อดสามารถ void หรือ return ค่าตัวแปรได้ทั้ง primitive หรือ object
- การส่งค่าตัวแปรผ่านพารามิเตอร์ ของเมท็อด (Parameter Passing) จาวาจะควบคุมการส่งผ่านพารามิเตอร์ ตามแต่ชนิดของข้อมูล user ไม่สามารถกำหนดเองได้
  - ถ้าส่งค่าตัวแปรประเภท primitive จะเป็นการ copy (pass by value)
  - ถ้าส่งค่าตัวแปรประเภท object ที่ถูกสร้างขึ้น(new) จะเป็นการ reference (pass by reference)
- Method ที่มีชื่อเหมือนกับ Class เรียกว่า Constructors
  - Constructor ต้องนำหน้าด้วย public และไม่มีการ void หรือ return แต่สามารถกำหนดให้มีพารามิเตอร์ตามที่ต้องการได้
  - Constructor ใช้ในการกำหนดการทำงานเริ่มต้นให้กับคลาส ถูกเรียกใช้อัตโนมัติเมื่อมีการสร้าง instance ของ class
- Method overloading
  - เมท็อดที่มีชื่อเดียวกัน แต่มีพารามิเตอร์ต่างกัน
  - เมื่อมีการเรียกใช้ javaจะเลือกเมท็อดที่มีพารามิเตอร์ตรงกันเท่านั้น ไปทำงาน
- การเรียก method ที่อยู่ต่างคลาสกัน และต่างไฟล์กัน จะต้องตั้งชื่อคลาสที่เรียกให้ตรงกับชื่อไฟล์ที่เก็บด้วย
- Java มี Static Methods ของคลาสต่างๆ ให้ผู้ใช้สามารถนำคำสั่งเหล่านั้นไปใช้ได้ โดยอ้างอิงผ่านชื่อคลาสนั้น โดยผู้ใช้ต้อง import คลาส จาก package ต่างๆ ที่ต้องการก่อน

Constructor

- **Constructor**
  - **Methods ที่ถูกตั้งชื่อเหมือนกับชื่อ class ขึ้นต้นด้วย public ไม่ต้องมี type**
  - **จะถูกเรียกใช้อัตโนมัติเมื่อมีการสร้าง instance ของ class นั้น (overload)**

```
public class Dnode {  // คลาสสำหรับข้อมูล 1 ตัว ชื่อ Dnode
   String word;     //Attribute ของ Dnode
   String mean;
   String type;
   public Dnode() {  // constructor สำหรับสร้างโหนดเปล่า เรียกใช้ด้วยคำสั่ง  Dnode x = new Dnode();
      word = "";
      mean = "";
      type = "";
   }
   public Dnode (String buff) { // ส่งข้อมูล 1 บรรทัดมาสร้างโหนด เรียกใช้ด้วยคำสั่ง Dnode x = new Dnode(buff);
      buff = buff.trim().replaceAll("\\s+"," "); {  //สมมุติ buff มีข้อมูลเป็น "str1,str2,str3" คั่นด้วย ","
      String[] str = buff.split(","); {  // แบ่งสตริงใส่ในอาร์เรย์ของสตริง โดยใช้ "," เป็นตัวคั่น
      word = str[0];        // นำข้อมูลที่แบ่งได้ไปเก็บ
      mean = str[1];
      type = str[2];
   }
   public Dnode (String str1,String str2,String str3) {//ส่งข้อมูลย่อยมาสร้างโหนด Dnode x = new Dnode(str1,str2,str3);
      word = str1;
      mean = str2;
      type = str3;
   }
}
```

Comparable & Comparator

- **เมื่อต้องการเรียกใช้ library ที่มีอยู่ เพื่อเรียงลำดับ และค้นหาข้อมูลแบบ binary**
  - **override method compareTo() ของ Comparable interface** // import java.Lang.Comparable
    - **สร้างคลาส T implements Comparable <T>**
    - **override method public int compareTo(T t)** //เปรียบเทียบ this.attr กับ t.attr return -1,0,+1
    - **สามารถสร้างวิธีเรียงลำดับได้ 1 วิธี**
  - **สร้างคลาส Comparator เพื่อเพิ่มวิธีการเรียงลำดับ** // import java.util.Comparator
    - **สร้างคลาส name implements Comparator<T> {**
    - **overide method public int compare(T t1, T t2)** //เปรียบเทียบ t1.attr กับ t2.attr return -1,0,+1
    - **สามารถสร้างได้มากกว่า 1 วิธีเพื่อเรียงลำดับเช่น เรียงตามรหัส เรียงตามชื่อฯลฯ**

```
class Dnode implements Comparable <Dnode> {
    String word;
    String mean;
    String type;
       public int compareTo (Dnode x) { //เรียงลำดับตาม word ส่งข้อมูลมาตัวเดียว เปรียบเทียบกับ this
              return (int) this.word.compareToIgnoreCase(x.word);
       }
 }
 class CmpWord implements Comparator <Dnode> {
      public int compare (Dnode x, Dnode y) {//เรียงลำดับตาม word ส่งข้อมูลมา 2 ตัว
                 return (int) x.word.compareToIgnoreCase(y.word) ;   }
}
 class CmpType implements Comparator <Dnode> {
     public int compare (Dnode x, Dnode y) {//เรียงลำดับตาม type
                 return (int) x.type.compareToIgnoreCase(y.type) ;   }
}
```

```
Collections.sort (data); //เรียงลำดับด้วย compareTo()
int i =Collections.binarySearch (data, key );
```

```
Collections.sort (data, new CmpWord() ); //เรียงลำดับด้วย compare()
i =Collections.binarySearch (data, key, new CmpWord() );
```

```
Collections.sort (data, new CmpType() );
i =Collections.binarySearch (data, key, new CmpType() );
```

Class & Instance

- แนวคิดในการเขียนโปรแกรมด้วยภาษาจาวา
  - สร้างคลาสสำหรับข้อมูล 1 ตัว (เทียบได้กับการกำหนดโครงสร้าง struct ของภาษาซี) และ methods หรือคำสั่งที่ช่วยในการจัดการข้อมูล 1 ตัว (เทียบได้กับการสร้าง functions) เช่น การใส่ค่า การหาคำตอบ การเปรียบเทียบเพื่อเรียงลำดับ การแสดงผล ฯลฯ
  - ถ้าต้องการจัดการกับข้อมูลจำนวนมาก ให้สร้างเป็นคลาสใหม่ โดยการสร้างคลาสใหม่อาจกำหนดเป็นอาร์เรย์ขึ้นมาเอง หรือใช้คลาสที่จาวาเตรียมไว้ให้ เช่น ArrayList ของคลาสย่อยนั้นๆ และสร้าง Method ที่ประมวลผลอาร์เรย์ในคลาสใหม่นั้น
- Instance คือตัวแปร (Object) ในภาษาจาวา มีชนิดเป็นคลาส ประกอบด้วยข้อมูล และคำสั่งของคลาสนั้น โดยถูกสร้างขึ้นด้วยคำสั่ง new
  - ตัวอย่างกรณีที่ผู้ใช้ต้องการสร้างอาร์เรย์เอง // สมมุติคลาสข้อมูลชื่อ Node

```
Node  [ ] data = new  Node [max];  // สร้าง data สำหรับใช้เป็นอาร์เรย์จำนวน max ตัว
        for (int i = 0; i<count; i++) // วนรอบเพื่อสร้าง instance ย่อยๆ จำนวน count ตัว
            data[i] = new Node() ; // สร้าง data[i] จำนวน count ตัว
```

    - จำนวนอิลิเมนต์ที่มีอยู่เท่ากับ max ตัว มีข้อมูลเก็บอยู่จริง count
    - ใช้ operator จัดการข้อมูลตำแหน่งที่ i ได้โดยตรง เช่น data[i] = x; x = data[i];
    - ต้องสร้างคำสั่งในการจัดการอาร์เรย์เองทั้งหมด เช่น การเพิ่ม การลบ การแทรกข้อมูล ฯลฯ
    - สามารถเรียกใช้คำสั่ง Arrays.<functions> มาช่วยประมวลผลได้ เช่น Arrays.sort()
  - ตัวอย่างกรณีที่ผู้ใช้เรียกใช้คลาสชื่อ ArrayList มาใช้ในการสร้างอาร์เรย์

```
ArrayList <Node> dataList = new ArrayList <Node> ; //ไม่ต้องกำหนดขนาด
        dataList.add(x) ; // ต้องเขียนวนรอบเพื่อเพิ่มข้อมูล x เข้าไปในอาร์เรย์ จนครบทุกตัว
```

    - จำนวนข้อมูลที่มีอยู่คือ data.size() //java จะปรับขนาดอาร์เรย์เท่าที่มีข้อมูลอยู่จริง
    - จัดการข้อมูลตำแหน่งที่ i ผ่านคำสั่ง เช่น x = dataList.get(i); dataList.set(i, x);
    - ใช้คำสั่งของคลาสที่เตรียมไว้ให้ในการจัดการอาร์เรย์ (หรือสร้างเพิ่ม)

Primitive Variables

- ตัวแปร Primitive Types มองข้อมูลเป็นตัวแปรทั่วไป

Type	Size	Default	Contains	C
boolean	1 bit	false	true or false	no
byte	8 bits	0	Signed Integer	yes
char	16 bits	0	Unicode(UTF-16) Character	ANSI 8 bits
short	16 bits	0	Signed Integer	yes
int	32 bits	0	Signed Integer	int/long/long int
long	64 bits	0	Signed Integer	long long
float	32 bits	0.0	IEEE754 floating point	yes
double	64 bits	0.0	IEEE754 floating point	yes

- ตัวแปรตัวเดียว สามารถจองตัวแปร ในบรรทัดที่ต้องการ แล้วใช้งานได้เลย

```
int  i = 0 ; // จองตัวแปร i และกำหนดค่าให้เป็น 0
```

- ถ้าเป็นอาร์เรย์ จะต้องสั่ง new เพื่อสร้าง ตัวแปร(pointer) ก่อน จึงจะใช้งานได้

```
int  [ ] data = new  int [100];   // data เป็น array ของ int  จำนวน 100 ตัว
      data[0] =  10 ;             // กำหนดค่าให้ตัวแปร data[]
```

- กรณีอาร์เรย์ 2 มิติ // กำหนดค่าให้ตัวแปร M[][]

```
double  M[][] = new double[10][10] ; // อาร์เรย์ 2 มิติ ขนาด 10x10
      M[0][1] =  1 ;
```

- ตรวจสอบขนาดของอาร์เรย์ที่จองไว้ด้วย .length เช่น M.length

Primitive Wrappers class

- **Wrapper Class ประกอบไปด้วยค่าของ ตัวเลข (primitive) และ เมท็อด**

Primitive type	Wrappers class	Constructor Arguments
boolean	Boolean	boolean or String
byte	Byte	byte or String
char	Character	char
short	Short	short or String
int	Integer	int or String
long	Long	long or String
float	Float	float, double or String
double	Double	double or String
	String	String

- **เป็นตัวแปรประเภท object จะต้องสร้างด้วยคำสั่ง new ก่อน(มี Constructor ที่รับข้อมูลได้ทั้งรูปแบบตัวเลขหรือ สตริง)**

```
Integer  i = new Integer(20) ;   // นำตัวเลข 20 มาสร้างตัวแปรจำนวนเต็ม i
Double  x = new Double("20.5"); // นำสตริง "20.5" มาสร้างตัวแปร x
```

- **มีคำสั่งในการแปลงข้อมูลที่เกี่ยวข้องกับคลาส ให้เรียกใช้**

```
int        i  =  Integer.parseInt("20"); // แปลงสตริงเป็นตัวเลข int
double   x  =  Double.parseDouble("20.5"); // แปลงสตริงเป็นตัวเลข double
String   str =  String.valueOf(20.5); // แปลงตัวเลขเป็น String
String   str =  w.toString();  //  ค่า String ของ Wrapper Class
```

Java Class : String

- String เป็นคลาสพิเศษ ของจาวา Java.lang.String ไม่จัดอยู่ใน primitive ไม่กำหนดความยาว
- สร้างตัวแปรแบบกำหนดค่าเริ่มต้น หรือผ่านคำสั่ง new

  String str = "CPE 113" ; // จองแบบกำหนดค่าเริ่มต้น

  String name = new String(str); // ใช้ Constructor สร้างข้อมูลเริ่มต้น

  String name = new String(); // ไม่กำหนดข้อมูลเริ่มต้น(null)

  บอกให้รู้ว่าเป็นอาร์เรย์ล่วงหน้า ไม่นิยมเขียน String str []
- ตรวจสอบจำนวนตัวอักษรที่เก็บอยู่ .length() เช่น str.length()
- กรณีที่ต้องการอาร์เรย์ของสตริง ต้องใช้คำสั่ง new ในการจองและกำหนดขนาดตัวแปรด้วย

  String [] strarry = new String[10]; // จองอาร์เรย์ของตัวแปรสตริง 10 ตัว str.length() = 10

  String [][] starry = new String[100][10]; // จองอาร์เรย์ของสตริง 2 มิติ

  // starry.length = 100 , starry[i].length = 10 , starry[i][j].length() = ความยาวของข้อมูล
- มีคำสั่ง split เพื่อสร้าง String ในอาร์เรย์ได้เอง

  String [] token = str.trim().split("\\s+");
- สามารถใช้ operator '=' ในการ copy และ '+' แทนการ concatenate ได้อย่างต่อเนื่อง

  str = "CPE"+"111"+" Programming with Data Structures;
- ตัวอย่างการแปลงค่าตัวเลขเป็น String
  - String str = String.ValueOf (123); // เรียกใช้เมท็อด ValueOf ของ String
- ตัวอย่างการแปลง String เป็นตัวเลข
  - int i = Integer.parseInt ("123"); // เรียกใช้เมท็อด parseInt ของ Integer
  - double d = Double.parseDouble("123.45"); // เรียกใช้เมท็อดของ Double
  - ถ้าแปลงไม่ได้จะเกิด error จะต้องมีการดักจับ error ด้วยคำสั่ง try {} catch{}

String Methods

- มีเมท็อดในคลาสสำหรับจัดการเกี่ยวกับสตริงโดยเฉพาะ เช่น
  - **length()** // int i = str.length();
  - **charAt()** // char ch = str.charAt(1) ;
  - **indexOf()** // int i = str.indexOf("E");
  - **compareTo , compareToIgnoreCase** // เปรียบเทียบ หรือใช้ .equal()

    ```
    int i = (str.compareTo("CPE113") ;
    ```
  - **isEmpty()** // boolean b = str.isEmpty() ;
  - **contains()** // boolean b = (str.contain("CPE") ;
  - **startsWith(),endsWith()** // boolean b = (str.endsWith("13") ;
  - **replace()** // แทนที่ข้อความ(regular expression) str = str.replace("\\+"," + ");
  - **trim()** // ตัดช่องว่างหน้าและหลังออก str = str.trim();
  - **toLowerCase() , toUpperCase()** // str = str.toLowerCase();
  - **split()** // แยกสตริงที่มีตัวคั่นใส่ในอาร์เรย์

    ```
    String [] token = str.split(" "); //ถ้ามี space 2 ตัวติดกัน จะตัดออกได้แค่ 1 ตัว
    ```
  - **StringTokenizer stok = new StringTokenizer(x, " ");** //ใช้ StringTokenizer

    ```
    token = new String[stok.countTokens()];
    for(int i=0; i<token.length; i++){
       token[i] = stok.nextToken();
    }
    ```
  - **format()** // str = String.format ("Answer = %d",i);

Standard input/output java

- จาวามี object ที่เป็น input/output มาตรฐานคือ //เหมือน stdin,stdout,stderr ของ C
  - System.in เป็น object ที่มี method สำหรับการอ่านข้อมูลทางอุปกรณ์ input มาตรฐาน คือ keyboard
  - System.out เป็น object ที่มี method สำหรับการแสดงข้อมูลออกทางอุปกรณ์ output มาตรฐาน คือ monitor
  - System.err เป็น object ที่มี method สำหรับการแสดงข้อผิดพลาดออกทางอุปกรณ์ที่ใช้ในการแสดงข้อผิดพลาด คือ monitor
- ตัวอย่างคำสั่งอ่านคีย์บอร์ด 1 ปุ่ม
  - int ch; ch = System.in.read(); //ให้ค่าเป็นตัวเลข ASCII ของปุ่มกด
- คำสั่งแสดงผลทางจอภาพ เรียกใช้ได้เลย ไม่ต้องสร้าง object
  - System.out.println([arg1][+arg2][+ .....]) ; // แสดงผลแบบต่อสตริง
    - ไม่ต้องมี format specifier พิมพ์ค่าสตริงหรือตัวแปรต่อเนื่องกัน โดยใช้ + เป็นตัวเชื่อม
    - ข้อมูลที่จะพิมพ์ถูกเปลี่ยนอยู่ในรูปแบบของสตริง สามารถพิมพ์ค่าตัวแปร primitive ได้โดยตรง //ถ้าต้องการพิมพ์คลาส ต้องสร้าง method toString() ไว้ในคลาสด้วย
    - ใช้ System.out.print ถ้าไม่ต้องการขึ้นบรรทัดใหม่
  - System.out.printf ("format", arguments); // แสดงผลแบบ format
    - รูปแบบ format ใช้เหมือนกับภาษาซี // %f =จำนวนจริง และ %d = จำนวนเต็ม
    - java แนะนำให้ใช้ %n(new line) แทน \n(line feed) //\n บางเครื่องใช้ไม่ได้

Scanner Class

- การอ่านข้อมูลจากคีย์บอร์ด จะอ่านผ่านคลาส Scanner โดยมี System.in เป็นพารามิเตอร์
- คลาส Scanner ใช้อ่าน Text Stream แล้วแปลงเป็นข้อมูล primitive หรือ String
- import java.util.Scanner
  - Scanner in = new Scanner (System.in); // in คือ ชื่อ instance ที่สร้างขึ้น
    - คลาส Scanner ใช้ในการอ่านข้อมูล เช่นตัวเลข หรือ สตริง จากคีย์บอร์ด โดยใช้ white space คั่นระหว่างข้อมูล
    - System.in คือ Input Stream (คีย์บอร์ด) ที่ต้องการอ่านจาก scanner
    - เรียกใช้ method ที่เกี่ยวข้องกับการอ่าน เช่น
      - in.next() , in.nextLine() อ่านสตริง 1 ตัว หรือ ทั้งบรรทัด
      - in.nextInt() , in.nextLong() อ่านจำนวนเต็ม 32 บิท หรือ 64 บิท
      - in.nextFloat(), in.nextDouble() อ่านจำนวนจริง 32 บิท หรือ 64 บิท
    - การอ่านค่าผิดรูปแบบสามารถทำให้เกิด error ได้ เช่นเดียวกับภาษาซี
    - มีเมท็อดตรวจสอบข้อมูลที่อ่านได้ (ก่อนเก็บเข้าตัวแปร) เช่น
      - in.hasNext() , in.hasNextInt(), in.hasNextDouble()
    - ไม่มีปัญหาในการอ่านตัวเลขที่เขียนผิดรูปแบบ เช่น 4x 1.1.1 //จะอ่านเป็นตัวเลขไม่ได้
    - เรียกใช้ in.nextLine() เพื่ออ่านค่าที่ค้างใน stream ทั้งหมดทิ้งได้
  - ต้องมีการเขียนโปรแกรมป้องกันความผิดพลาดที่อาจจะเกิดขึ้นได้ Error handling

Error Exception handling

- เราสามารถใช้คำสั่ง try { ..... } catch (Exception ...) { ..... } เพื่อจัดการความผิดพลาดที่เกิดขึ้นได้

ของเขตคำสั่งที่จะดักจับความผิดพลาด | ชนิดของ Error | ชุดคำสั่งที่จะทำเมื่อเกิด Error

```
static double Read_Double(String msg) {
   double a = 0.0;
   boolean  success = false;
   Scanner in = new Scanner(System.in); //สร้าง object ของ scanner ชื่อ in เพื่ออ่านคีย์บอร์ด
   while (!success) {// วนรอบขณะที่อ่านไม่สำเร็จ
              System.out.print(msg); //แสดงคำถามที่ส่งมา
              try {
                        a = in.nextDouble(); //อ่านข้อมูลที่เป็น double ใส่ใน a
                        success = true; // ถ้าอ่าน a สำเร็จ ทำต่อ success
             } catch (Exception e) {  // ถ้าอ่าน a ไม่สำเร็จจะกระโดดมาทำ catch
                  System.out.println("Error ! Please Enter number again.");
                  in.nextLine();  //อ่านข้อมูลที่ค้างทิ้งทั้งหมด
               }
   }
   in.close();
   return a;
}
```

ถ้าอ่านแล้วเกิดความผิดพลาดโปรแกรมจะกระโดดไปที่ Excpetion

```
double   a = Read_Double("Enter number of a ") ;
```

```
Enter number of a 5X ↵
Error ! Please Enter number again.
Enter number of a _
```

Error Exception handling

- **เราสามารถใช้คำสั่ง try { ..... } catch (Exception ...) { ..... } เพื่อจัดการความผิดพลาดที่เกิดขึ้นได้**

```
static int Read_Int(String msg, int min, int max) {
int a = 0;
boolean success = false;
Scanner in = new Scanner(System.in);
            while (!success) {
                  try {
                        System.out.print(msg);
                        a = in.nextInt();
                        if (a >= min && a <= max)
                           success = true;
                        else
                           System.out.printf("Please enter between %d - %d\n", min, max);
                  } catch (Exception e) {
                        System.out.println("Error Please Enter number again.");
                        in.nextLine();
                    }
            }
            return a;
}
```

```
int   a = Read_Int("Enter number between 0-50 ",0,50) ;
```

```
Enter number between 0-50  500 ↵
Please enter between 0 - 50
Enter number between 0-50 _
```

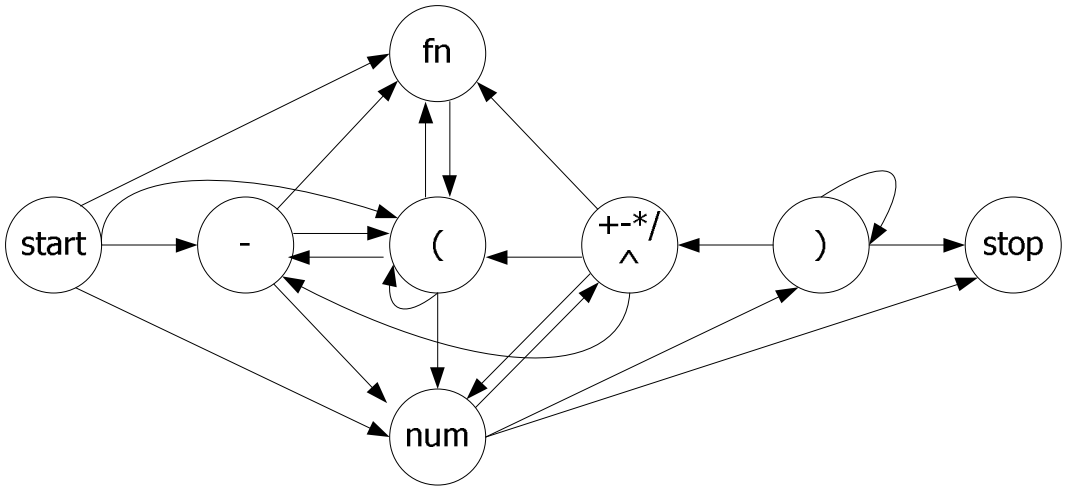
Assignment 5.1 Check Token

- โจทย์ปัญหา กำหนดฟังก์ชัน main() ที่ใช้ในการทดสอบสตริงไว้แล้ว ให้สร้างฟังก์ชันที่สอดคล้องกับการเรียกใช้ และให้ผลลัพธ์ตามที่กำหนด
- ขั้นตอน เขียนฟังก์ชันทดสอบสตริงที่สอดคล้องกับการเรียกใช้และผลลัพธ์
  - แยกสตริง ใส่ในอาร์เรย์
  - number คือตัวเลขจำนวนจริงเช่น 12.5 123
  - function ได้แก่คำว่า sin,cos,tan,asin,acos,atan,sqrt,log,exp,abs รวม 10 ตัว
  - operator ได้แก่เครื่องหมาย + - * / ^ ( ) รวม 7 ตัว
  - variable/constant ได้แก่ คำที่ตั้งชื่อไว้ล่วงหน้า เช่น ans , pi, g , e เป็นต้น
  - error ได้แก่ คำอื่นๆ ที่นอกเหนือจากที่กำหนด
- ตัวอย่างฟังก์ชันที่ควรจะสร้างขึ้น
  - สร้างฟังก์ชันแบ่ง token ใส่ในอาร์เรย์ return count //อาจต้องเพิ่ม space
  - สร้างฟังก์ชันตรวจสอบ negative sign (- , ! , ~)
  - สร้างฟังก์ชันตรวจสอบชนิดของ operator return operator group/code
  - สร้างฟังก์ชันตรวจสอบตัวเลขจำนวนจริง return success/value
  - สร้างฟังก์ชันตรวจสอบชื่อตัวแปร (ans, pi, g, e) return success/value
  - สร้างฟังก์ชันตรวจสอบชื่อเฉพาะ (function) return function group/code
  - ฟังก์ชันก์ในการตรวจสอบลำดับ syntax และ ( )
  - กรณีฟังก์ชันทำงาน return code ให้กำหนด ไม่สำเร็จให้เป็น 0 สำเร็จ >0

Assignment 5.1

- **Syntax Group**
  - **(เครื่องหมายวงเล็บเปิด**
  - **) เครื่องหมายวงเล็บปิด**
  - **function ได้แก่ sin,cos,tan,asin,acos,atan,sqrt,log,exp,abs รวม 10 ตัว**
  - **negative sign เครื่องหมาย [-] ที่เขียนนำหน้าตัวเลข(อาจใช้ตัวอื่นแทน)**
  - **binary operator ได้แก่เครื่องหมาย [+ - * / ^] รวม 5 ตัว (3 ลำดับความสำคัญ)**
  - **number/variable ได้แก่ตัวเลขจำนวนจริง,ชื่อตัวแปร ans,ค่าคงที่ pi**
- **Balance bracket และต้องไม่มี) มาก่อน (**
- **State diagram**

	Present	Next State
0	start	(, fn, number, sign
1	number	operator,) , stop
2	operator	number, fn, (, sign
5	sign	number, fn, (
6	fn	(
7	(	(, fn, number, sign
8	)	operator,) , stop
9	stop	



Assignment 5.1 Test Case

1. expression> 2.5+pi
answer> OK
2. expression> 2.5 pi + // postfix
answer> error
3. expression> -2.54 + 1.68 (2 + 3) // ตัวเลข ตามด้วย (
answer> error
4. expression> -2.5 - sin (-30) // Negative sign
answer> OK
5. expression> sin(cos(tan(asin(acos(atan(log(sqrt(exp(abs(ans)))))))))//วงเล็บปิด 9 ตัว
answer> error
6.expression> sin(cos(tan(asin(acos(atan(log(sqrt(exp(abs(ans))))))))))//วงเล็บปิด10 ตัว
answer> OK
7. expression> 2 + 3 - ((4))
answer> OK
8. expression> 2 + 3-(4/5*) // operator ตามด้วยวงเล็บ)
answer> error
9. expression> 2 + 3-(4/5*6+()) // วงเล็บ(ตามด้วยวงเล็บ)
answer> error
10. expression> 2 + 3)+4/5*6+((7) // วงเล็บปิด) มาก่อน วงเล็บเปิด(
answer> error
11. expression> -1+2-3*4/5^6
answer> OK
12. expression> help // หรือใช้คำสั่ง token
answer> token = sin, cos, tan, asin, acos, atan, sqrt, log, exp, abs +, -, *, /, ^, (,), pi, ans
expression> end // คำสั่งจบโปรแกรม
End program

Assignment 5.1 Check Syntax

```
public class TokenAnalysis {
.....
public static void change_negative_sign(String token[]) {
   int i;
     if (token[0].equals("-"))
         token[0] = "!";
     for (i = 0; i < token.length - 1; i++)
         if (token[i + 1].equals("-") && (token[i].matches("[-+/(*\\^]")))
            token[i + 1] = "!";
}
public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner in = new Scanner(System.in);
  String str;
      do {
              System.out.print("Input String>>");
              str = in.nextLine(); อ่านข้อมูล 1 บรรทัด
              str = Transform(str);  //แปลงรูปสตริงให้สามารถ split ได้
              System.out.println(str);
              String[] token = str.split(" "); // แยก สตริงใส่ใน token[] โดยใช้เครื่องหมายเว้นวรรคคั่น
              change_negative_sign(token); //เปลี่ยนเครื่องหมาย negative sign - ให้กลายเป็น !
              int state = check_state(token); //เช็คความถูกต้องของประโยคคำสั่ง
              if (state == 1 || state == 8) //สถานะที่ถูกต้อง
                 System.out.println("Syntax OK\n");
              else
                 System.out.println("Syntax Error\n"); //ไม่สามารถจบประโยคได้
      } while (!str.equalsIgnoreCase("end"));
      System.out.println("End Program");
      in.close();
}
}//end class
```

Assignment 5.1 Check String

- รับข้อมูล String จาก คีย์บอร์ด 1 บรรทัด
  Scanner in = new Scanner(System.in);
  String str = in.nextLine();
- เพิ่ม/ลด/แทรก space ระหว่าง operator
- แยกสตริงใส่ในอาร์เรย์ของสตริง โดยใช้ " "เป็นตัวคั่น
  String [] token = str.trim().split("\\s+");
- ตรวจสอบตัวเลข
- ตรวจสอบชื่อฟังก์ชัน
- ตรวจสอบชื่อตัวแปร

```
public static String Transform(String x) {
   x = x.replace("+", " + ");
   x = x.replace("-", " - ");
   x = x.replace("*", " * ");
   x = x.replace("/", " / ");
   x = x.replace("^", " ^ ");
   x = x.replace("(", " ( ");
   x = x.replace(")", " ) ");
   x = x.replaceAll("\\s+", " "); //ปรับช่องว่าง
   x = x.trim(); //ตัดช่องว่าง หน้า หลัง
   return x;
}
```

```
public static boolean isNumber(String str) {
    boolean check = true;
     try {
            Double.parseDouble(str);
     } catch (Exception e) {
            check = false;
     }
      return check;
}
```

```
public static boolean isConstant(String str) {
    String[] myconst = { "pi", "e", "ans" };
    boolean ans = false;
      for (int i = 0; i < myconst.length; i++) {
           if (myconst[i].equalsIgnoreCase(str))
              ans = true;
      }
   return ans; // true/false
}
```

```
public static int check_Function(String str) {
 String[] myfunc = { "", "sin", "cos", "tan", "cosec", sec",
 "cot", "asin", "acos", "atan", "sqrt",  "log", "exp", "abs" };
 int ans = 0;
    for (int i = 1; i < myfunc.length; i++) {
        if (myfunc[i].equalsIgnoreCase(str))
            ans = i; // รหัสของ function
    }
    return ans; // 0 คือ error
}
```

Assignment Check String

- ตรวจสอบกลุ่ม
- ตรวจสอบ State diagram

```
public static int check_state(String[] token) {
int state = 0, next = 0, b = 0, i;
  for (i = 0; i < token.length && state >= 0; i++) {
    state = next; //เริ่มต้น state ใหม่
    next = check_group(token[i]);
    if (next >=2 && next <= 4) next = 2 ; // + - * / ^ คือ state 2
    if (next == 0)  state = -1; // คำที่ไม่รู้จักเป็นให้ error
    if (next == 7)  b++; //check (
       else if (next == 8) b--; //check )
    if (b < 0)
      state = -1; // error จาก ) มากกว่า หรือมาก่อน (
    else if (state == 0 && (next == 8 || next == 2))
      state = -1; // ขึ้นต้นด้วย )  -   +
    else if (state == 1 && (next == 1 || next == 6 || next == 7))
      state = -1; // ตัวเลข ตามด้วย ตัวแปร, fn หรือ (
    else if (state == 2 && (next == 2 || next == 8))
      state = -1; // operator ตามด้วย operator หรือ )
    else if (state == 5 && (next == 2 || next == 8))
      state = -1; // เครื่องหมาย ตามด้วย operator หรือ )
    else if (state == 6 && (next != 7))
      state = -1; // function ไม่ได้ตามด้วย (
    else if (state == 7 && (next == 8 || next == 2))
      state = -1; // ( ตามด้วย  ) หรือ operator
  } //วนรอบครบทุกตัว
 if (b != 0 || state == -1) // b = การครบคู่ของวงเล็บ
     return -1;
 else
     return next; // return สถานะสุดท้าย
}
```

```
public static int check_group(String str) {
    if (isNumber(str))
        return 1;
    else if (isConstant(str))
        return 1;
    else if (str.matches("[-+]"))
        return 2;
    else if (str.matches("[/*]"))
        return 3;
    else if (str.equals("^"))
        return 4;
    else if (str.equals("!"))
        return 5;
    else if (str.equals("("))
        return 7;
    else if (str.equals(")"))
        return 8;
    else if (check_Function(str)>0)
        return 6;
    else
        return 0; // คำที่ไม่รู้จัก
}
```