Java 정리

OOP(Object-oriented Program) – object, class, packages, interfaces, inheritance

Object: real-world object(literally anything): can have two characteristics(state and behaviour) – state to be data and behaviour to be action

States in private and behaviours in public

# What is a program?

Program = a sequence of instructions for a computer to follow

Primitive programming languages

* Machine language and assembly language

High level programming languages

* Human friendly programming languages

Interpreter – translates and executes each command alternatively

* Translates everytime the program runs
* Interactive

Compiler – translates the whole or a part of the program into machine code or another language code(except java, C# etc.)

* Compile once execute often
* Fast

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

What is a virtual machine(VM)?

* Virtual computer running on your real computer
* JVM
  + A virtual computer that executes only java program(java bytecode)
    - NoGui
    - Command line program

그니깐 compiler가 .java를 byte-code인 .class로 만든다 JVM을 위해서. JVM이 Machine code로 넘겨줘서 컴퓨터가 읽는다.

장점은 compile once, run everywhere

단점은 performance overhead다.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

.java를 javac \*.java로 compile

\*.class는 java classname으로 실행

Java development kit(jdk)는 java compilers, jvm, 다른 libraries와 commands가 포함되어있다.

Package edu.hanndong.csee.java.bikes

Public class Bicycle{ - a class name must be the same as the java source code file name

Modifiers

* Public
* Protected
* No modifier
* Private

Non-access modifiers:

* Static
* Final
* Abstract
* Violate

Compile

* + $ javac -cp [path to jar or existing class files for reusing] -d [path to save your .class [path/to/java files]\*.java [path/to/java files]\*.java
  + Example  
    javac -d ./bin src/edu/handong/csee/java/l04/\*.java (Unix/Linux, Mac shell)  
    javac -d .\bin src\edu\handong\csee\java\l04\\*.java (Windows CMD)

Javac -d bin src/myfirstjava/test.java – compile

$ javac -cp /path/to/MyJar.jar -d bin /src/your/package/\*.java

Java -cp bin myfirstjava.test – 실행

To disassemble bytecode (javap -c classfile)

* Disassemble
  + Translate (a program) from machine code into a higher-level programming language.

Main method

Public static void main(String args[])

The main method accepts a single argument: an array of elements of type String (the args)

* + $ java -cp [class path] com.myapckage.MyClass arg1 arg2

Javadoc

Put Javadoc comments right before class, field, and method definitions. Note that only definitions with the 'public' modifier will be shown in Javadoc!!

Format: javadoc -d [path-to-save-javadoc] [source-file-names]

e.g. $ javadoc -d javadoc-directory edu/handong/csee/java/mypackagename/Test.java

Jar file

* To distribute our classes to other developers.
* Command  
  $ jar -cvf Sugang.jar -C /path/to/your/package/path .  
  e.g., jar -cvf MyJar.jar -C bin .

Errors  
Syntax error

* Syntax error: if you mistype part of a program, the compiler may issue a syntax error. Or omitting necessary parts such as semi colon
* Sometimes the compiler can’t guess your intent and prints a confusing error message or multiple error messages if the error cascades over several lines

Semantic error

* The compiler warns you each time you use a variable that has not been initialized

Runtime problems

* Classpath 설정 안돼서 못읽는 경우

# OOP concepts

Benefits of Coding based on objects

* Modularity
* Information-hiding
* Code reuse
* Pluggability and debugging ease

Class is a blueprint of an object (design plan of the object)

* Objects: concept
* A class: detailed specification of objects
* An instance: a real object created based on the class.

Inheritance:

Bicycle class 아래에 산악자전거 길거리 자전거, 2인 자전거 등등을 상속클래스라고 함

* 그래서 Bicycle class는 superclass 자전거 종류들은 subclass들이다.

보통 상속을 할때는

Class roadbicycle extends Bicycle{} 해준다

Bicycle class를 사용함은 물론 subclass만의 특성도 해줄 수 있다.

Interface

* Contact point to interact with other objects
* A heard of a method without body in java
* 클래스들이 가져야 할 기본 메서드가 저장되어있음
* 이 interface 를 쓸때에는 클래스에 안에서 interface가 가지고 있는 메소드를 모두 구현해야함
* 이유: interface forms a contract between the class and the outside world, and this contract is enforced at build time by the compiler.
* Mechanism to guarantee clearer and safter program design and implementation
* 선언 public interface name{}
* 쓸때는 class 이름 옆에 implements name 해준다

Package

* A namespace that organizes a set of related classes and interfaces.
* A class library = a set of packages
  + API(application programming interface)
  + A set of reusable classes

# Class more

super(startCadence, startSpeed, startGear);

위 코드는 subclass가 superclass를 construct할 때 쓰이는 코드이다.

To create a new Bicycle object(an instance) called myBike

* Use the new operator
* Bicycle myBike = new Bicycle(30,0,8)

The compiler automatically provides a no-argument, default constructor for any class without constructors.

class MyClass extends MySuperClass implements YourInterface

this.x 는 class의 field 변수이다

Encapsulation – private field들은 다른 class에서 사용이 안돼서 getter 와 setter로 사용가능

Method 를 선언 후 부를 때 method signature라는 것을 사용해서 선언 methodname(pararmeters)

이름을 선언할 때 verb부분은 소문자 나머지는 대문자 첫글자만

Overloading methods

* Methods의 이름은 같으나 parameters를 달라서 자바가 구별할 수 있는 경우

Object

* Create an object (instance) from a class
  + **Declaration**: The code set in **bold** are all variable declarations that associate a variable name with an object type.
  + **Instantiation**: The new keyword is a Java operator that creates the object.
  + **Initialization**: The new operator is followed by a call to a constructor, which initializes the new object.
* public class CreateObjectDemo {
* public static void main(String[] args) {
* // Declare and create a point object and two rectangle objects.
* **Point originOne** = new Point(23, 94);
* **Rectangle rectOne** = new Rectangle(originOne, 100, 200);
* **Rectangle rectTwo** = new Rectangle(50, 100);

The new operator returns a reference to the object itcreated

Garbage collector

* No worries to destroy obejcts in java
* The java runtime environment(JRE) deletes objects when it determines that they are no longer being used => Garbage collection

Annotations

* Provide data about a program that is not part of the program itself
* Information for the compiler 컴파일러에게 추가적인 정보제공 (@Override)
* Compile-time and deployment-time processing 시간 배분
* Runtime processing – 이때 처리

Packages

You should bundle these classes and the interface in a package for several reasons:

* + You and other programmers
    - can easily determine that these types are related.
    - know where to find types that can provide graphics-related functions.
  + The names of your types won't conflict with the type names in other packages
    - because the package creates a new namespace.
  + You can allow types within the package to have unrestricted access to one another
    - yet still restrict access for types outside the package.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

The prefix java.awt (the Java Abstract Window Toolkit) is used for a number of related packages to make the relationship evident, but not to show inclusion.

Importing java.awt.\* imports all of the types in the java.awt package, but *it does not import java.awt.color, java.awt.font, or any other java.awt.xxxx packages*.

Returning a value from method

* can return
  + a value of a primitive type
  + a reference type
  + a class or interface

Access level

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Use private unless you have a good reason not to

* Class members that are defined by the 'static' modifier.
  + Class variables
  + Class methods

Sometimes, you want to have variables that are common to all objects of a class -> use the static modifier. Without constructing or stating class you can use the method.

Final 은 const랑 같은 역할 변경시 compie-time error 발생

A compile-time constant: If a primitive type or a string is defined as a constant and the value is known at compile time, the compiler replaces the constant name everywhere in the code with its value; - have to recompile any classes that use this constant to get the current value.

* Naming for variables
  + Case-sensitive, unlimited length of Unicode letters and digits, beginning with a letter, $, or \_.
    - But begin with a letter (convention)
    - White space is not permitted.
  + Subsequent characters may be letters, digits, dollar signs, or underscore characters.
    - Use full words instead of cryptic abbreviations. e.g., gear
  + For only one word, spell that word in all lowercase letters.
  + For more than one word, capitalize the first letter of each subsequent word. e.g., gearRatio, currentGear
  + For constant: e.g., static final int NUM\_GEARS = 6

Int[] arr = new int[5]

System.arraycopy(copyFrom, 2, copyTo, 0, 7);

copyFrom 2에서부터 copyTo0부터 7번째까지

char[] copyTo = java.util.Arrays.copyOfRange(copyFrom, 2, 9);

equals로 비교가능 array간 또는 string도’

instanceof() – 특정 object로 됐는지 확인

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Java.io.PrintStream

System.out.format(“%f”,num);

A new line character appropriate to the platform running the application. You should always use %n rather than \n

tB – a date and time

td, te – a date and time conversion -2-digit dayof month. Td leading zeros while te does not

java.text.\* to use string.format() or new DecimalFormat

DecimalFormat myFormatter = new DecimalFormat(pattern);

String output = myFormatter.format(value);

Math in java.lang but math is all in static so you can call it as Math.pow()

* + int number = (int)(Math.random() \* 10); // random between 0 and 9

Boolean endsWith

Boolean startsWith

compareToIgnoreCase

Autoboxing – 기본값을 참조형으로

Unboxing – 참조형을 기본값으로 Integer.valueOf(값)

Recursion

* use more storage space than iterative version
  + overhead during runtime
* also runs slower
* However in some programming tasks, recursion is a better choice, a more elegant solution
* Merge sort – a recursive sorting method
  + A divide-and-conquer algorithm
    - Araray to be sorted is divided in half
    - The two halves are sorted by recursive calls
    - This produces two smaller, sorted arrays which are merged to a single sorted array
* View [Java implementation](about:blank), listing 11.8, class MergeSort
* public static void sort (int [] a)
* {
* if (a.length >= 2) {
* int halfLength = a.length / 2;
* int [] firstHalf = new int [halfLength];
* int [] lastHalf = new int [a.length - halfLength];
* divide (a, firstHalf, lastHalf);
* sort (firstHalf);
* sort (lastHalf);
* merge (a, firstHalf, lastHalf);
* }
* //else do nothing. a.length == 1, so a is sorted.
* }

Interfaces

* A contract that spells out how software developed by different groups of programmers interacts

We should avoid changes in interface

* If you make changes, then all classes that implement the old DoIt interface will break because they no longer implement the old interface.
* Programmers relying on this interface will protest loudly
* Option 1: Design well first
* Option 2: creating a DoItPlus interface that extends DoIt
* Option 3: can define your new methods as default methods
  + It will enable you to add new functionality to the interfaces of your libraries and ensure binary compatibility with code written for older versions of those interfaces 그니깐 굳이 interface implement한다고 새로운 파일에 꼭 필요한 건 아니다. 만약 필요하다면 쓰면 되는거.

Summary

* interface declaration can contain method signatures (abstract methods), default methods, static methods and constant definitions.
* The only methods that have implementations are default and static methods.
* A class that implements an interface must implement all the methods declared in the interface.
* An interface name can be used anywhere a type can be used.

마지막 꺼 의미

* interface Animal {
* void sound();
* }
* class Dog implements Animal {
* public void sound() {
* System.out.println("Woof");
* }
* }
* class Cat implements Animal {
* public void sound() {
* System.out.println("Meow");
* }
* }
* public class Main {
* public static void main(String[] args) {
* Animal animal1 = new Dog(); // 인터페이스를 변수의 자료형으로 사용
* Animal animal2 = new Cat(); // 인터페이스를 변수의 자료형으로 사용
* animal1.sound(); // Dog 객체의 sound 메소드 호출
* animal2.sound(); // Cat 객체의 sound 메소드 호출
* }
* }

Nested class

* 다른 클래스 내부에 정의된 클래스를 의미함
* 외부 클래스의 멤버 변수 및 메소드에 쉽게 접근 가능
* 코드 구조를 조직화하고 캡슐화하여 코드의 가독성과 유지 보수성을 향상시키는데 사용됌
* 크게 4가지
  + Static nested class: 외부 클래스의 instance에종속되지 않는 정적 클래스. 외부 클래스의 인스턴스에 접근 불가하며, 주로 관련된 유틸리티 클래스나 독립적으로 사용 (static은 사용가능 근데 그냥 field는 불가)
  + Intner class: 외부 클래스의 인스턴스에 종속되는 클래스, 외부 클래스의 멤버에 접근 가능
  + Local inner class: 메소드나 블록 내부에 정의되는 클래스로, 해당 메소드나 블록 내부에서만 접근 가능
  + Anonymous inner class: 이름 없는 내부 클래스로, 주로 인터페이스나 추상클래스를 구현할 때 사용
* OuterClass outerObject = new OuterClass();
* OuterClass.InnerClass innerObject = outerObject.new InnerClass();

Anonymous inner class 예

Public class Outer {

Interface MyInterface{

Void doSomething();

}

MyInterface anonymousInner = new MyInterface() {

@Override

Public void doSomething(){}

}

}

Why nested classes

* **It is a way of logically grouping classes that are only used in one place.**
  + **Used as helper classes**
* **It increases encapsulation.**
* **It can lead to more readable and maintainable code.**

**Lambdab**

* **Useful when an interface contains only one method**
* **public class Calculator {**
* **interface IntegerMath {**
* **int operation(int a, int b);**
* **}**
* **public int operateBinary(int a, int b, IntegerMath op) {**
* **return op.operation(a, b);**
* **}**
* **public static void main(String... args) {**
* **Calculator myApp = new Calculator();**
* **IntegerMath addition = (a, b) -> a + b; // implementing for the interface method, operation, by using an anonymous method.**
* **IntegerMath subtraction = (a, b) -> a - b;**
* **System.out.println("40 + 2 = " +**
* **myApp.operateBinary(40, 2, addition));**
* **System.out.println("20 - 10 = " +**
* **myApp.operateBinary(20, 10, subtraction));**
* **}**
* **}**

**}**

**public class Operators {**

**public static int add(int a, int b) {**

**return a+b;**

**}**

**public static int sub(int a, int b) {**

**return a-b;**

**}**

**}**

**----**

**System.out.println("41 + 3 = " + myApp.operateBinary(41, 3, Operators::add));**

**Abstract methods and classes**

* **An abstract method is a method that is declared without an implementation.**
* **public abstract class GraphicObject {**
* **private int size;**
* **public void getSize() {**
* **return size;**
* **}**
* **public void setSize(int size) {**
* **this.size = size;**
* **}  
    
   // an abstract method**
* **public abstract void draw();**
* **}**

**By using abstract methods from an abstract class, we can enforce the subclass of this abstract class to implement abstract methods**

**추상 클래스를 사용하는 경우:**

**1. 여러 가까이 관련된 클래스 간에 코드를 공유하고 싶을 때.**

**2. 추상 클래스를 확장하는 클래스가 많은 공통 메소드나 필드를 가지거나, public 이외의 접근 제어자(protected나 private와 같은)가 필요할 때.**

**3. 정적이거나 final이 아닌 필드를 선언하고 싶을 때. 이를 통해 객체의 상태를 수정하고 접근하는 메소드를 정의할 수 있습니다.**

**인터페이스를 사용하는 경우:**

**1. 관련 없는 클래스가 인터페이스를 구현할 것으로 예상될 때. 예를 들어, Comparable이나 Cloneable과 같은 인터페이스는 관련 없는 다양한 클래스에서 구현됩니다.**

**2. 특정 데이터 타입의 동작을 지정하고자 하지만, 동작을 구현하는 클래스에 관심이 없을 때.**

**3. 타입의 다중 상속을 활용하고자 할 때.**

**만약 class X가 interface Y를 implement한다고 했을 때, interface의 모든 메서드를 구현해야하지만 하지 못했을 경우 class X는 abstract으로 되어야 한다. 하지만 만약 class XX가 extends X를 하는 클래스가 Y interface의 남아있는 메서드를 모두 구현한다면, class XX는 abstract으로 선언되지 않아도 된다.**

**Enum types:**

* **Enum type is a special data type that enables for a variable to be a set of predefined constants. 미리 데이터 형태를 선언하는 것**
* **public enum Day {**
* **SUNDAY, MONDAY, TUESDAY, WEDNESDAY,**
* **THURSDAY, FRIDAY, SATURDAY**
* **}**

**Overriding is rewriting the method that is already written but using same method signatures.**

**Hiding is to rewrite static method in subclass that is already written in super class.**

**Cat myCat = new Cat();**

**Animal myAnimal = myCat;**

**Animal.testClassMethod();**

**myAnimal.testInstanceMethod();**

**The static method in Animal**

**The instance method in Cat**

**Hiding은 어떤 class로 객체가 생성됐냐에 따라 다름 반면 overriding은 객체 생성과 상관없이 어떤 것을 지금 가지고 있냐 따라 다름**

* **Interface Methods**
  + ***Default* methods and *abstract* methods in interfaces are inherited like instance methods. However, when the supertypes of a class or interface provide multiple default methods with the same signature, the Java compiler follows inheritance rules to resolve the name conflict.**
  + **These rules are driven by the following two principles:**
    - **Instance methods are preferred over interface default methods.**
    - **Methods that are already overridden by other candidates are ignored. This circumstance can arise when supertypes share a common ancestor.**

**Polymorphism – 같은 이름의 메소드나 연산자가 여러 클래스에서 다른 방식으로 동작할 수 있는 능력을 의미함.**

**메소드 오버로딩과 메소드 오버라이딩으로 나눌 수 있음**

**public void printDescription() {**

**super.printDescription();**

**System.out.println("The " + "MountainBike has a" +**

**getSuspension() + " suspension.");**

**}**

**Override 됌**

**Clone()**

**public class MyClass implements Cloneable {**

**// 필드와 메서드들...**

**@Override**

**protected Object clone() throws CloneNotSupportedException {**

**return super.clone(); // 얕은 복사**

**// 깊은 복사를 위한 추가적인 코드 작성도 가능**

**}**

**}**

**public class Book {**

**...**

**public boolean equals(Object obj) {**

**if (obj instanceof Book)**

**return ISBN.equals((Book)obj.getISBN());**

**else**

**return false;**

**}**

**}**

**.equals()**

**Finalize() – 가비지 컬렉터에 의해 객체가 소멸되기 직전에 호출됌**

* **메서드를 사용하여객체의 정리 작업을 수행**
* **사용 권장 X**

**getClass() – 자바의 모든 클래스에서 상속되는 object클래스에 정의된 메서드이며 객체의 클래스 정보를 반환 class이름 반환 .getSimpleName()과 같이 사용**

* **hashCode() -** The value returned by hashCode() is the object's hash code, which is the object's memory address in hexadecimal.