CSC207 Week 2

Larry Zhang

Today’s Outline

- Finish up non-OO examples, Array, HashMap
- OO programming
- Unit Test: JUnit (if time permits)
- Javadoc (if time permits)

Recap

- Java code is compiled into bytecode which runs in JVM.
- Java uses **static** typing (main difference from Python)
  - All variables must be declared with a type before being used.
  - The type of the variable CANNOT be changed after being declared.
- Primitive types:
  - byte, short, int, long, float, double
  - char
  - boolean
- Class types (all other types, provided by Java or user defined)
  - e.g., String, Integer, Double, Balloon, Person...

Auto-Conversion Rules

- “can-auto-convert” directions:
  - byte → short → int → long → float → double
  - char → int and above
  - boolean → no other types
- Any other conversion that do not following the “can-auto-convert” directions must be explicitly **casted**.
  - e.g., `float x = (float)2.07;`
  - `long x = 207; short y = (short)x;`
  - `short x = 207; long y = x;` // auto-converted, no need for casting

Understanding the Auto-Conversion Rules

NEVER just memorize the rules! Understand the reasoning of behind the rules!

- Reasoning #1 (technical):
  - Converting from **shorter** to **longer** types just involves adding 0's to the original number, without loss of information
    - e.g., converting **byte x = 7** to a **short int** is adding eight 0's at the higher bits to the original number:
    - **0000 0111** (byte) → 0000 0000 **0000 0111** (short)
  - Converting from longer to shorter types requires **truncating** the original number, may lose information
    - e.g., convert **short x = 259** to a **byte** is removing the highest eight 0's from the original number:
    - 0000 0001 **0000 0011** (short value 259) → **0000 0011** (byte value 3)

Understanding the Auto-Conversion Rules

Reasoning #2 (conceptual and more general, not just for primitive types):

- If Type A **is-a** special case (subtype) of Type B, then it is okay to convert from A to B.
- Example: Cat is a special case of Animal, so it is fine to convert an instance of Cat into type Animal.
  - “This is a cat” implies “this is an animal”
- Conversely, converting an instance of Animal into type Cat might not be fine.
  - “This is an animal” does not imply “this is a cat”.

Sidenote: IS-A, HAS-A, RESPONDS-TO

In Object-Oriented Design (OOD), we often use the above three phrases to describe the relationships between different classes.

- **IS-A**: the “subtype” relationship
  - Example: A car is a vehicle.
- **HAS-A**: the “composition” relationship
  - Example: A car has a wheel.
- **RESPONDS-TO**: communication (message passing) between objects.
  - Example: A person P calls a car C’s **C.openDoor()** method, C responds to P.

These phrases are part of the **Universal Modelling Language (UML)**.

DEMO

Non-OO Code Examples Cont.

`ControlsFunctions.java`

(if statements, loops, functions, Array, HashMap)

Object Oriented Programming

Object Oriented Programming in Java

- We have done OO programming in Python, so you know what it’s all about.
- So let’s focus on the differences between Python and Java.
- Overall, Java is a more rigorous language than Python in terms of expressing the structure of OO design.

Constructor

```
public Balloon(String color) {
    // "this" is like Python's "self"
    this.amount = 0;
    this.capacity = 100;
    this.popped = false;
    this.color = color;
}
```

In Python, it is a method named “__init__”

In Java, it is a method with the same name as the class.

- has no return type, not even void.
- can have multiple constructors with different type signatures.
    - the one with the matching signature will be called
- “**this**” is like Python’s “**self**”

```
public Balloon(String color, int capacity) {
    // "this" is like Python's "self"
    this.amount = 0;
    this.capacity = capacity;
    this.popped = false;
    this.color = color;
}
```

Access Modifiers

In Java, each variable and method of a class has a clearly declared access modifier which defines from where you can access this variable/method. This is access control is enforced by the compiler (violation will cause compiler error). Python doesn't have this.

A package is a namespace which contains several classes which are supposed to be in the same "package", e.g., "java.lang.Math", "java.utils", or user-defined "csc207"

Read more:
http://docs.oracle.com/javase/tutorial/java/concepts/package.html

Modifier	Class	Package	Subclass	World
`public`	Yes	Yes	Yes	Yes
`protected`	Yes	Yes	Yes	No
`default (package private)`	Yes	Yes	No	No
`private`	Yes	No	No	No

Encapsulation

How do we decide the access modifiers of the variables and methods?

Lazy idea: Let everything be public. What could go wrong?

Better Idea: Only make public the methods that provide the “interface” to the users of the class.

```
class TimHortons:

    variables:
        coffeeMachine
        beans
        oven
        bread
        lights
        …

    methods:
        orderFood()
        orderCoffee()
        payMoney()
        grindBeans()
        toastMuffin()
        addMilk()
        addSugar()
        getFood()
        turnOffLights()
        …
```

keyword: static

- With static, it is a **function**, which is called by name only; it is independent of any object.
- Without static, it is a **method**, which is called by a name that is associated to an object.

```
public class Person {

  // function
  public static int plusOne(int x) {
    return x + 1;
  }
}
```

```
public class Person {

  private int age;

  // method
  public void becomeOlder() {
    this.age += 1;
  }
}
```

keyword static used on variables

- It is a variable which belongs to the **class** and **not to object** (instance)
- Static variables are initialized **only once** , at the start of the execution.
- A **single copy** to be shared by all instances of the class

```
public class Person {
    public static int nPeople = 0; // doesn't have to be public

    // Constructor
    public Person() {
        nPeople++; // keeps track of # of Person's constructed
        System.out.println(nPeople + " objects created");
    }
}
// … outside the class, static variables can be accessed as follows
System.out.println(Person.nPeople);
```

DEMO

Translate “Balloon.py” to “Balloon.java”
“UseBalloon.java”

Java is pass-by-VALUE

For primitive-typed variables, the **VALUE of the variable** (not the reference of the variable) is passed to a function.

For class-typed variables, the **VALUE of the reference** to the is passed to function.

Example 1

```
public static void main(String args[]) {
    int p1 = 42;
    func(p1);
    System.out.println(p1); // what's the value of p1
}

public static void func(int x) {
    x = 7;
    // this does not change the value of p1
    // because x is just a COPY of the value of p1
}
```

Example 2

```
Balloon p1 = new Balloon("red");
func(p1);
System.out.println(p1); // what's the amount and color of p1?

public static void func(Balloon b) {
    // add air to the balloon at the address stored in b
    b.addAir(42);
    b = new Balloon("yellow");
    // b is changed to storing the address of the yellow balloon,
    // BUT this does not change the value of p1
    // because b is just a COPY of the value of p1
}
```

Reading: More on Pass-by-Value

http://javadude.com/articles/passbyvalue.htm

UML

- Unified Modeling Language (UML) allows us to express the design of a program before writing any code.
- It is language-independent.
- An extremely expressive language.
- We'll use only a small part of the language, Class Diagrams, to represent basic OO design.

Notation

Data members:

```
name: type
```

Methods:

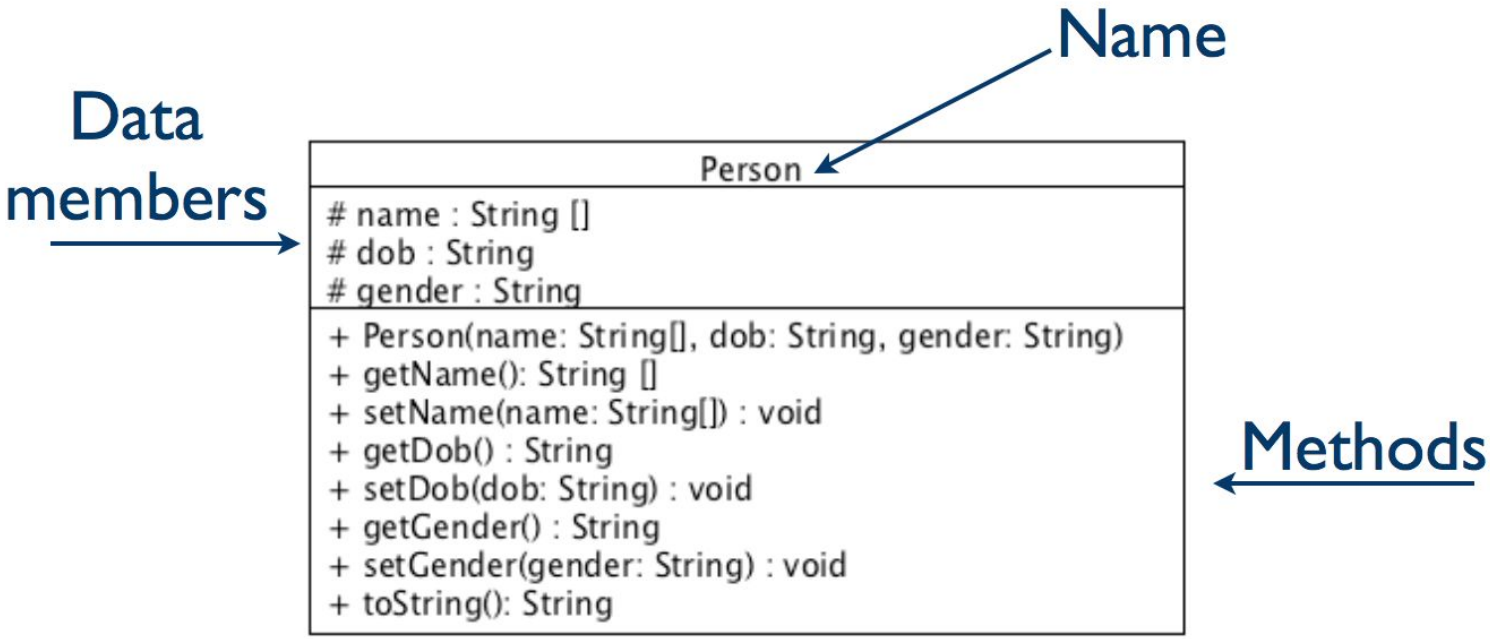
```
methodName(param1: type1, param2: type2,... ): returnType
```

Visibility:

- \- private
- \+ public
- \# protected
- ~ package

Static: underline

Example: Class Person



References

- Java OO:
  - http://docs.oracle.com/javase/tutorial/java/javaOO/index.html
- Access Modifiers:
  - http://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html
- JUnit
  - http://junit.org/
- Javadoc
  - http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html
- Kahoot quiz can be found by search “CSC207” on Kahoot.com