



CMSC 131

Object-Oriented Programming I

Classes Introduction IV

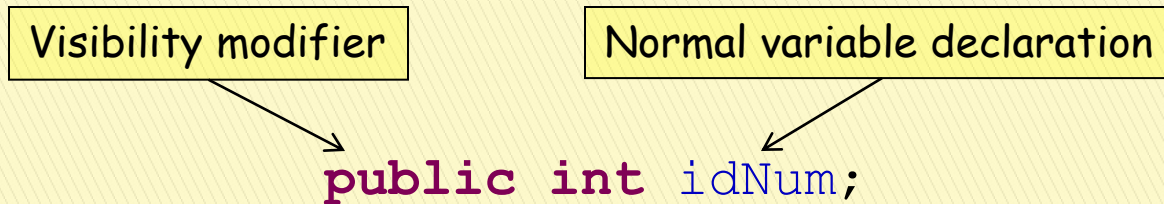
**Dept of Computer Science
University of Maryland College Park**

This material is based on material provided by Ben Bederson, Bonnie Dorr,
Fawzi Emad, David Mount, Jan Plane

Overview

- ▶ Instance data
- ▶ Static vs. Non-static methods
- ▶ Constructors

Anatomy of an Instance Variable Declaration



- ▶ We will see later that we can have private as visibility modifier
 - ▶ If you don't specify any modifier then it is considered package
- ▶ What is the default value?
 - ▶ Boolean → false, Number → 0, Reference → null
- ▶ **null** → represents "no address"
- ▶ These variables (unlike local variables) are visible from all the methods of the class!
- ▶ Let's draw a diagram
- ▶ **Example:** Person.java

Object Creation

- ▶ Once a class is **defined**, objects based on that class can be created using new: ***new Student()***
- ▶ We are “creating an instance of class Student”
- ▶ To assign an object to a variable, the variable’s type must be the class of the object
`Student s = new Student();`
- ▶ Each object has its own copies of all the instance variables in the class
- ▶ Instance variables and methods in an object can be accessed using “.” or using setter (mutator) methods

`s.idNum = 123456789;`
`s.setIdNum(123456789);`

Two Types of Methods (Static/NonStatic)

- ▶ So far we have seen static methods
- ▶ **Static methods can be called** by using the name of the class followed by a period

ClassName.staticMethodName()

- ▶ In a class a static method can call another static method without having to specify the class name (what we have been doing in our previous examples)

Two Types of Methods (Static/Non-Static)

- ▶ **Non-Static methods** are methods that **need an object** in order for them to be called. They are designed to use the data associated with an object, therefore they **need an object in order to be called**
- ▶ Unlike static methods, you cannot call a non-static method via the class name. You need to use an object reference

objectRef.nonStaticMethodName()

- ▶ Static vs. Non-Static methods:
 - **Example:** StaticVsNonStatic.java
- ▶ Static methods can only call static methods
- ▶ Non-Static methods can call static and non-static methods
- ▶ **When to define a method as static?**
 - **When it makes no reference to object data (instance data)!**

Constructors

- ▶ Special “methods” in class definitions to specify how objects are initialized
- ▶ Constructor is a bad name
 - They should have been called initializers
 - **new** operator creates the object
- ▶ **They have as name the name of the class**
 - They don't have return type
- ▶ You don't call them; **they are called for you!**
- ▶ Possible onstructor definition for Student class

```
Student (String nameDesired, int idDesired) {  
    name = nameDesired;  
    id = idDesired;  
}
```

Constructors

- ▶ Can have more than one constructor provided argument lists are different

```
Student (int IdDesired) {  
    id = IdDesired;  
}
```

- ▶ Constructors with no parameters is called the default constructor

```
Student () {  
    ...  
}
```

- ▶ **Example:** PersonWithConstructors.java

About Default Constructor

- ▶ Important→ If you don't define a constructor you get the default constructor;
- ▶ If you define any constructor (no matter which one) you will not get the default constructor java provides. You will need to define it yourself.
- ▶ **Example:** DefaultConstExampleA.java,
DefaultConstExampleB.java