Non-static features of classes

Last week, we discussed static features of classes

* Static features use the keyword “static”
* Static features have to do with the class itself – not with any particular instance
* If a class contains only static features, it is called a utility class – basically, utility classes are “toolboxes”

Today we will start introducing non-static features of classes

* All classes contain three things:
	+ Methods – the bits of code that manipulate data, in particular, the data stored in the object
	+ Fields – the data contained within the object
	+ Constructors – special methods that are executed when the object is created – normally, they give values to the object’s fields
		- In a utility class, we have an empty constructor set to private
		- Even if you don’t provide a constructor for a class, Java provides one by default
		- Constructors always have the same name as the class and no return type – also, capitalization matters

Example: BankAccount class

Scenario: Your company is given the job of creating software for a major bank to handle their electronic banking system. As part of this job, you must write code to handle bank accounts, according to a given API.

Methods and constructors can be defined multiple times with different parameters. Java decides which one you want to use based on the parameters you provide.

Scope of variables

Variable names are defined within the nearest braces

int a;

{ int a;}

Need to specify which one you mean, if there is more than one

Keyword this: “this” refers to the current object that you are working on

e.g. in a constructor:

public BankAccount(double balance)

{

 this.balance = balance;

}

in this example, “this” refers to the object being created; this.balance refers to the balance field of the object being created

this() called as a method will call a constructor

If you add a non-default constructor, you can no longer use the default constructor