CONSTRUCTORS

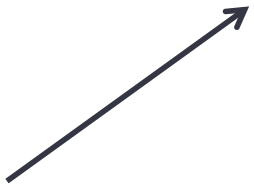
CS302 – Introduction to Programming
University of Wisconsin – Madison
Lecture 19

By Matthew Bernstein – matthewb@cs.wisc.edu

Constructors

- A **constructor** initializes the instance variables of an object
- The constructor is automatically called when an object is created with the **new** operator
- Example:

```
Scanner scan = new Scanner(System.in);
```



scan's constructor is called when it is created with the **new** operator

Constructors

- The name of the constructor MUST be identical to the class name
- The constructor MUST NOT define a return type
- The constructor MUST be public
- Example:

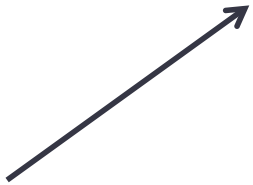
```
class Car
{
        String make;
        int mileage;

        // constructor
        public Car()
        {
                make = “Ford”;
                mileage = 0;
        }
}
```

Constructor Parameters

- Constructors can be passed arguments
- Example:

```
Scanner scan = new Scanner(System.in);
```



System.in is passed as an argument to the constructor

Defining Constructors

- Let's say we want to define the Car's make and model when this object is created. We can define the Car's constructor such that it accepts a String corresponding to the Car's make and a String corresponding to its model.

Constructor Example

```
class Car
{
        // Instance variables
        private String make;
        private String model;
        private int mileage;

        // This constructor initiliazes the make and model
        // to its parameter values
        public Car(String k, String d)
        {
                make = k;
                model = d;
        }
}
```

Deciding Constructor Parameters

- How do we decide which instance variables we will initialize in a class's constructor and which we will leave as the default value?
- In general, if an object *requires* some instance variable to be set to a specific value *before* the user uses this object, then you should initialize it in the constructor
- For example, the Scanner object needs to know where the input is coming from. We supply it with System.in so that our Scanner object knows to read data from the keyboard.

Deciding Constructor Parameters

- In general, deciding which instance variables should be initialized in the constructor is a design choice
- The best practice is to make sure that the user's program won't crash when she calls any of the object's methods simply because the object was not initialized correctly

Constructors are different from Methods

- Constructors are *similar* to methods in that they correspond to a segment of code that is executed when called, but DO NOT think of them as methods
- An object's constructor is called only once (when the object is created) and can never be called again
- You cannot call an object's constructor. It is only called when the object is created

Constructor Overloading

- You can define multiple constructors for your class
- This is called **constructor overloading**
- The constructor that is called when the object is created will depend on the arguments you pass to the constructor

Constructor Overloading Example

```
class Car
{
        private String make;
        private String model;
        private double originalPrice;

        // First Constructor
        public Car(double op)
        {
                originalPrice = op;
        }

        // Second Constructor
        public Car(String k, String d)
        {
                make = k
                model = m
        }
}
```

Calling Either Constructor

- Now when we create an instance of Car, we can either instantiate it with the first constructor or the second constructor:

- Either:

```
double price = 23000;

// First constructor
Car carA = new Car( price );
```

- Or

```
String make = “Ford”;
String model = “Mustang”;

// Second constructor
Car carA = new Car(make, model);
```

Default Constructor

- If we do not define a constructor for our class, a constructor is still called. It looks like this:

```
public Car()
{ }
```

- This is called the **default constructor**. It is called when you do not explicitly define a constructor.
- When you define a constructor, the default constructor is "overwritten"

Common Design Practice: Using **this** to call a primary constructor

- If we define multiple constructors, it is good design practice to make one of these constructors the “primary” constructor
- We then make a call the “primary” constructor from the “non-primary” constructor using the **this** reserved word

Example

```
class Car
{
        private String make;
        private String model;
        private double originalPrice;

        // The “primary” constructor
        public Car(String make, String model, double originalPrice)
        {
                this.make = make;
                this.model = model;
                this.originalPrice = originalPrice;
        }

        // Other constructors then call the primary the constructor
        public Car(double originalPrice)
        {
                this(null, null originalPrice);
        }

        // This correlates to the default constructor
        public Car()
        {
                this(null, null, 0);
        }
}
```

Cool Link

- What is Information Theory?
- http://www.youtube.com/watch?v=p0ASFxKS9sg&list=SPbg3ZX2pWlgKDVFNwn9B63UhYJVIerzHL