# Classes & Objects

Ch. 7

## Recall

- We've used objects in the past – of type **String, Scanner, File**
- These are data types that serve special and unique purposes
- We can invent our own data-types that have special data and actions, and create objects of that data-type

## A simple data type

- What are some data that all students have?
- What are some actions that all students can do?

## Student Class

# Elements of a simple class

- A class describes data that belongs to an object, and operations that object can perform
- **Instance Variables** – The data belonging to the class. aka fields
- **Instance Methods** – The methods belonging to the class. aka member methods

```
public class Person{
   int age;
   String name;
   double height;
}
```

In general, your class should be kept in a separate java file

```
public class PersonTest{
   public static void main(String[] args){
      Person p = new Person();
      p.age = 25;
      p.name = "Jerry";
      p.height = 70.5;
      printPerson(p);
   }
   public static void printPerson(Person p){
      System.out.println("Age: " + p.age);
      System.out.println("Name: " + p.name);
      System.out.println("Height: " + p.height);
   }
}
```

```
// What is printed by this program?

public class PersonTest{
   public static void main(String[] args){
      Person p = new Person();
      p.age = 25;
      p.name = "Jerry";
      p.height = 70.5;
      makeOlder(p);
      printPerson(p);
   }
   public static void printPerson(Person p){
      System.out.println("Age: " + p.age);
      System.out.println("Name: " + p.name);
      System.out.println("Height: " + p.height);
   }
   public static void makeOlder(Person q){
      q.age++;
   }
}
```

```
public class Person{
   int age;
   String name;
   double height;
}
```

## Constructors

- Used when we want to initialize our instance variables during the same step we initialize our objects
- Will always have the same name as our class
- If we do not create a constructor, we get a default constructor

## Overloading Constructors

- Similarly to how we overloaded methods, we can overload constructors
  - The Constructor name remains the same (it will always be the same name as the class)
  - The parameters of constructors will be different

# Exercise

- Write a method for our Student class called **equals** that determines if two student objects are equal (as in, all of their instance variables are the same), and returns true/false (hint: parameter is Student object; method returns boolean value)

## Exercise

- **Write a class** named BankAccount with the following:
  - Instance variables: **balance** (double), **name** (String), **currency** (String) − this will refer to USD, CAD, JPY, or any other currency of the world.
  - Instance methods: **update balance** (changes object's balance), **print balance** (prints out balance), **close account** (sets balance to zero)
  - Constructors: One constructor that only initializes name, another that initializes all 3 instance variables