# Lecture 12

# Data Types and Strings

# Class v. Object

- A Class represents the generic description of a type.
- An Object represents a specific instance of the type.
- Video Game=>Class, WoW=>Instance
- Members of a class (variables, methods) can only accessed through an instance, UNLESS they are defined as STATIC.

# When to use Static

- Static – this allows you to reference something from the class, not an instance (object).
  Main

  Attributes (class variables)

  - Constants like PI or GRAV (9.81 m/s/s)

  Methods that don't use object attributes

  - Currency.convertDollarsToYen()

- Makes sense if you have a method or attribute that you might use outside the existence of an object.
- Used with 'Utility' classes – classes that define useful constants or methods but don't store things, like 'Math'.
- Can factor out some often used code

# Primitive v. Reference Variables

- Primitive variables actually contain the value that they have been assigned.

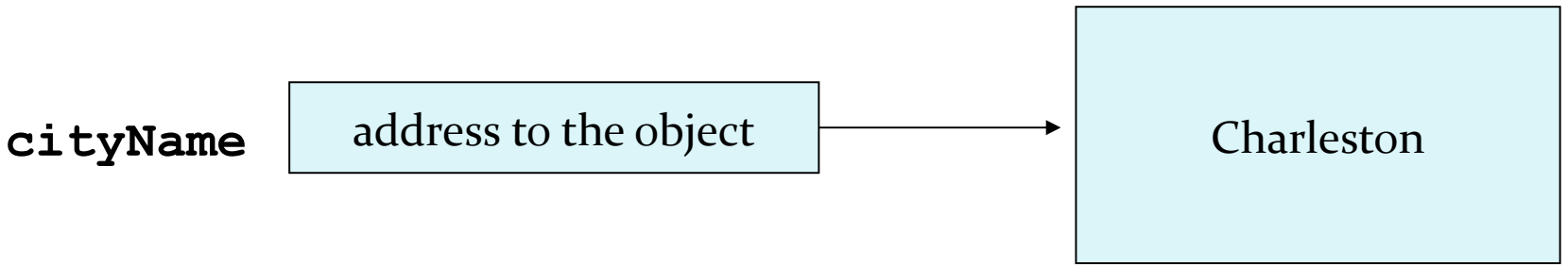
```
int number = 25;
```

- The value 25 will be stored in the memory location associated with the variable **`number`**.
- Objects are not stored in variables, however. Objects are *referenced* by variables.
- (Remember, this is why an array of Objects contains Null pointers until each of those objects is initialized. An array of a primitive, like Ints, contains a value '0'.)

# Primitive v. Reference Variables

- When a variable references an object, it contains the memory address of the object's location.
- Then it is said that the variable *references* the object.

```
String cityName = "Charleston";
```




The object that contains the character string "Charleston"

# The String class

- Java has no primitive data type that holds a series of characters.
- The **String** class from the *Java Standard Library* is used for this purpose.
- In order to be useful, the a variable must be created to reference a **String** object. `String number;`
- Notice the **S** in **String** is upper case.
- By convention, class names should *always* begin with an upper case character.

* neat fact – String is often based on an array of characters

# String objects

- A variable can be assigned a string **literal**.

```
String value = "Hello,World!";
```

- **String** objects are the only objects that can be created in this way.
- A string **variable** can be created using the ***new*** keyword.

```
String value = new String("Hello, World!");
```

Here **value** string can be changed into a different string by Interacting somehow with the code in the programs

```
value = "Goodbye, World!";
```

This is the method that all other objects must use when they are created.

# String Methods

- Since **String** is a class, objects that are *instances* of it have methods.
- One of those methods is the **length** method.

```
stringSize = name.length();
```

- This statement calls the length method on the object pointed to by the **name** variable.
- All the methods can be found by looking at the on-line Java docs.

# String equals v. ==

- The String class over-rides the .equals() method to return true if the two strings have the same content.
- The == method returns true of the two string variables refer to the same place in memory.

# Enum Types

An enum type is a special data type that enables for a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it. Common examples include compass directions (values of NORTH, SOUTH, EAST, and WEST) and the days of the week.

Because they are constants, the names of an enum type's fields are in uppercase letters.

In the Java programming language, you define an enum type by using the enum keyword. For example, you would specify a days-of-the-week enum type as:

```
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY
}
```

* enum types are the kind of thing that is often static

# Using Enum Types

```
public class EnumTest {
    Day day;
    public EnumTest(Day day) {
        this.day = day;
    }
    public void tellItLikeItIs() {
        switch (day) {
            case MONDAY:
                System.out.println("On Mondays, I never go to work.");
                break;
            Case TUESDAY:
                System.out.println("On Tuesdays, I stay at home.");
                break;
            case FRIDAY:
                System.out.println("Fridays are better.");
                break;
            case SATURDAY: case SUNDAY:
                System.out.println("Practice all day on Saturday, on Sundays I play best.");
                break;
            default:
                System.out.println("Practice trumpet every day");
                break;
        }}}
```