Object Oriented Programming – SCJ2153

Classes from API: String, Scanner, JOptionPane

Associate Prof. Dr. Norazah Yusof

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 name;
```

- Notice the `S` in `String` is upper case.
- By convention, class names should always begin with an upper case character.

`String` Objects

- A variable can be assigned a String literal.

```
String ucapan1 = "Selamat";
```

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

```
String ucapan2 = new String("Sejahtera");
```

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

`String` Objects

- Example:

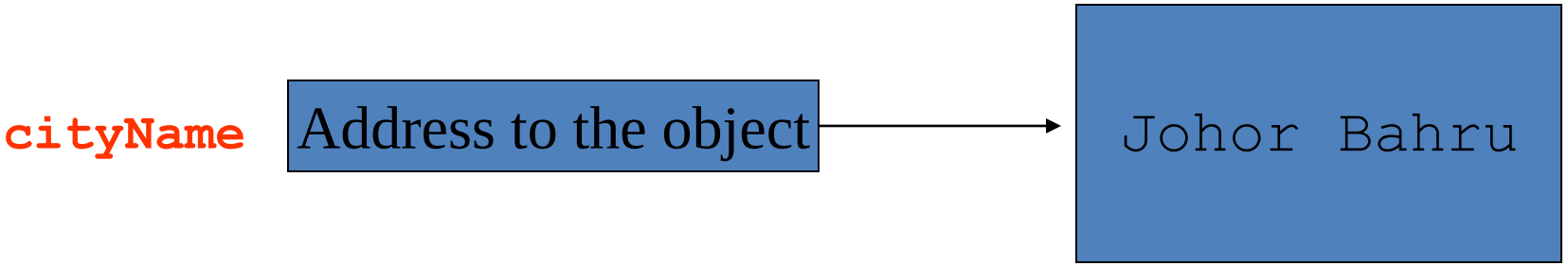
```
public class StringDemo {
 public static void main(String[] args) {
   String text1 = "Happy";
   String text2 = new String("Holiday");

   System.out.print(text1+" ");
   System.out.println(text2);
 }
}
```

Reference Variables

- Objects are not stored in variables, however. Objects are *referenced* by 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 = "Johor Bahru";
```

The object that contains the character string "Johor Bahru"



The `String` Methods

- Since `String` is a class, its objects have methods.
- One of those methods is the `length` method, which returns an integer value.
- Example:

```
String cityName = "Johor Bahru";
int stringSize;
stringSize = cityName.length();
```

- The above statements run the `length` method on the object pointed to by the `cityName` variable.

The `String` Method: `length()`

- Example:

```
public class StringDemo1 {
 public static void main(String[] args) {
   String myMessage = "All the best";
   String name = "Farhana";
   int messageSize;
   messageSize = myMessage.length();
   System.out.println(" The length of "+ myMessage
                      +" is " + messageSize);
   messageSize = name.length();
   System.out.println("The length of "+name+" is "
                      +messageSize);
 }
}
```

The `String` Method: `toLowerCase()`, `toUpperCase()` and `charAt()`

- Example:

```
public class StringDemo2 {
 public static void main(String[] args) {
   String place = "Johor Bahru";
   String upper = place.toUpperCase();
   String lower = place.toLowerCase();
   char aLetter = place.charAt(2);
   int stringSize = place.length();

   System.out.println(place);
   System.out.println(upper);
   System.out.println(lower);
   System.out.println(aLetter);
   System.out.println(stringSize);
 }
}
```

The `Scanner` Class

- To read input from the keyboard we can use the `Scanner` class.
- The `Scanner` class is defined in `java.util`, so we will use the following statement at the top of our programs:

```
import java.util.Scanner;
```

The `Scanner` Class

- Scanner objects work with `System.in`
- To create a `Scanner` object:

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

- Use appropriate `Scanner` class method to read the data from the keyboard and assign the value to a variable:

```
int num1 = keyboard.nextInt();

double num2 = keyboard.nextDouble();
char letter = keyboard.next();
String word = keyboard.nextLine();
```

The `JOptionPane` Class

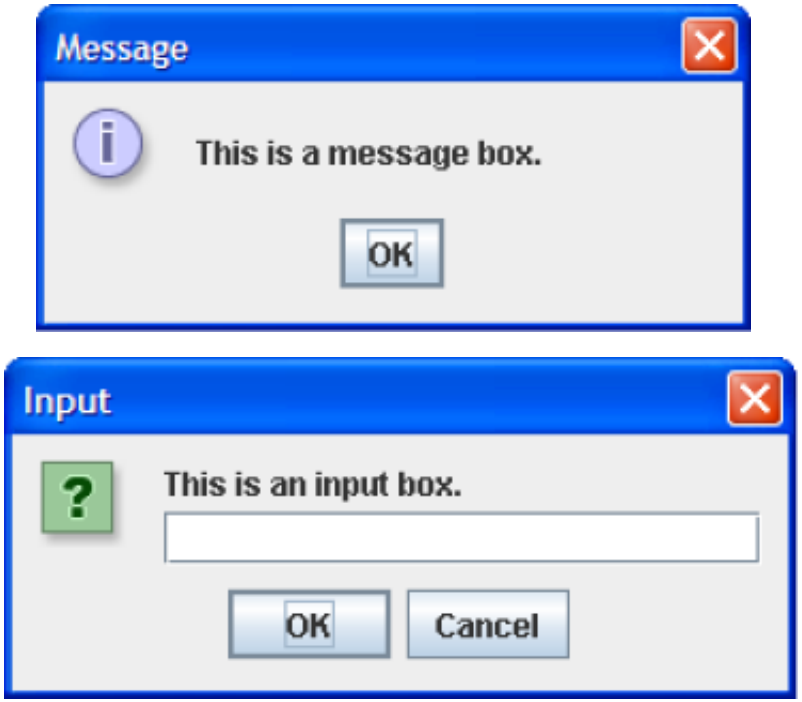
- The `JOptionPane` class is not automatically available to your Java programs.
- The following statement must be before the program's class header:

```
import javax.swing.JOptionPane;
```

- This statement tells the compiler where to find the `JOptionPane` class.

The `JOptionPane` Class

The `JOptionPane` class provides methods to display each type of dialog box.



Message Dialogs

- `JOptionPane.showMessageDialog` method is used to display a message dialog.

```
JOptionPane.showMessageDialog(null, "Hello World");
```

- The second argument is the message that is to be displayed.

Input Dialogs

- An input dialog is a quick and simple way to ask the user to enter data.
- The dialog displays a text field, an Ok button and a Cancel button.
- If Ok is pressed, the dialog returns the user's input.
- If Cancel is pressed, the dialog returns null.

Input Dialogs

```
String name;
name = JOptionPane.showInputDialog(
          "Enter your name.");
```

- The argument passed to the method is the message to display.
- If the user clicks on the OK button, `name` references the string entered by the user.
- If the user clicks on the Cancel button, `name` references `null`.

The `System.exit` Method

- A program that uses `JOptionPane` does not automatically stop executing when the end of the main method is reached.
- Java generates a *thread*, which is a process running in the computer, when a `JOptionPane` is created.
- If the `System.exit` method is not called, this thread continues to execute.

The `System.exit` Method

- The `System.exit` method requires an integer argument.

```
System.exit(0);
```

- This argument is an *exit code* that is passed back to the operating system.
- This code is usually ignored, however, it can be used outside the program:
  - to indicate whether the program ended successfully or as the result of a failure.
  - The value 0 traditionally indicates that the program ended successfully.

Converting a String to a Number

- The `JOptionPane`'s `showInputDialog` method always returns the user's input as a `String`
- A `String` containing a number, such as "127.89, can be converted to a numeric data type.

The Parse Methods

- To convert a string to a number, need to use the numeric wrapper classes:
  - The `Integer` class has a method that converts a string to an `int`,
  - The `Double` class has a method that converts a string to a `double`
- Example: to read an integer with an input dialog, need to use this conversion technique:

```
int number;
String str;
str = JOptionPane.showInputDialog("Enter a
number.");
number = Integer.parseInt(str);
```