Lecture (03)
String Variables

Dr. Ahmed ElShafee

Agenda

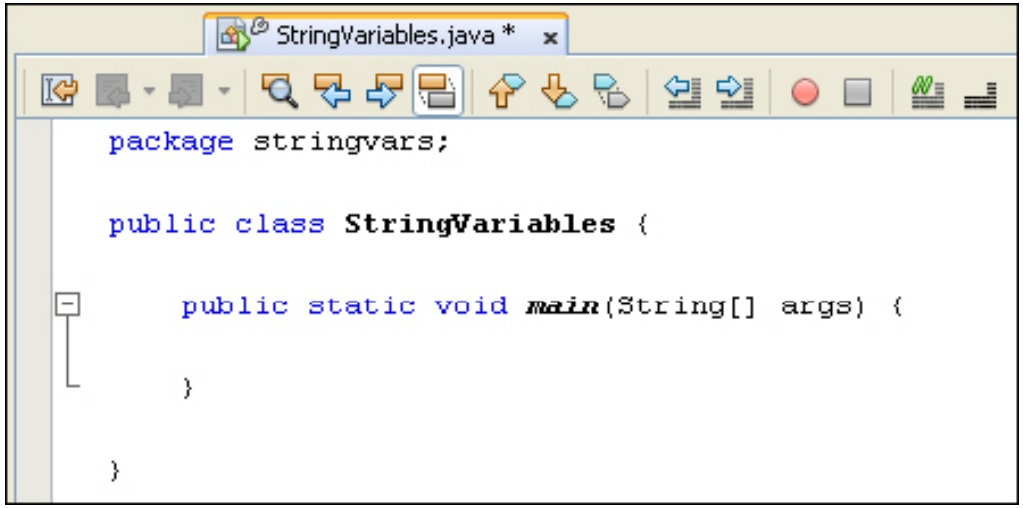
- Introduction
- StringVars example
- StringVars02 example
- Accepting Input from a User
- StringVars3
- Option Panes (GUI)
- StringVar04
- Exercises

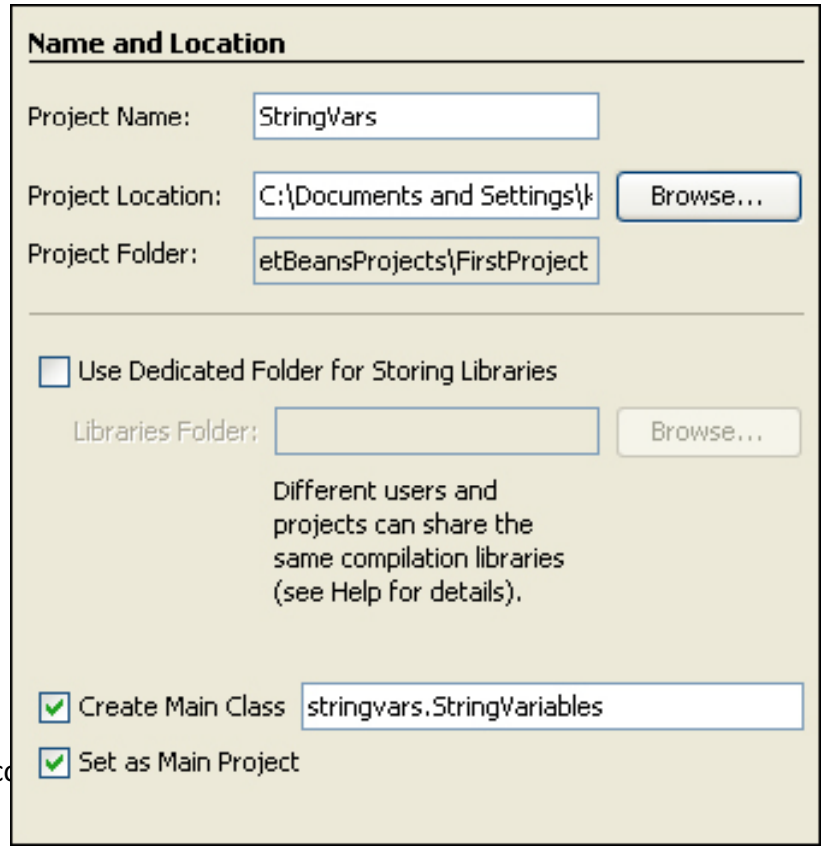
Introduction

- As well as storing number values, variables can hold text. You can store just one character, or lots of characters.
- To store just one character, the **char variable is** used.
- Usually, though, you'll want to store more than one character.
- To do so, you need the **string variable type.**

StringVars example

- Start a new project from NetBeans.
- Make sure **Java and Java Application are selected.**
- Project Name is StringVars, and the Class name is StringVariables.





- To set up a string variable, you type the word **String followed by a name for your** variable.
- Note that there's an uppercase "S" for String. Again, a semicolon ends the line:

String first_name;

- Assign a value to your new string variable by typing an equals sign.
- After the equals sign the text you want to store goes between two sets of double quotes:

first_name = "Ahmed";

- If you prefer, you can have all that on one line:

String first_name = "Ahmed";

- Set up a second string variable to hold a surname/family name:

String family_name = “Soliman”;

- To print both names, add the following println( ):

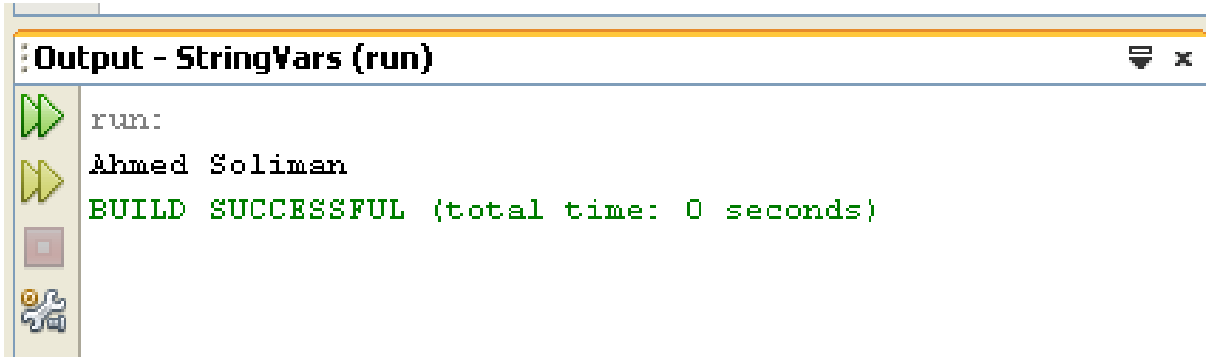
System.out.println(first_name + " " + family_name);

- In between the round brackets of **println( ), we have this:**

first_name + " " + family_name

```
public class StringVariables {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        String first_name = "Ahmed";
        String family_name = "Soliman";
        System.out.println(first_name + " " + family_name);
    }
}
```

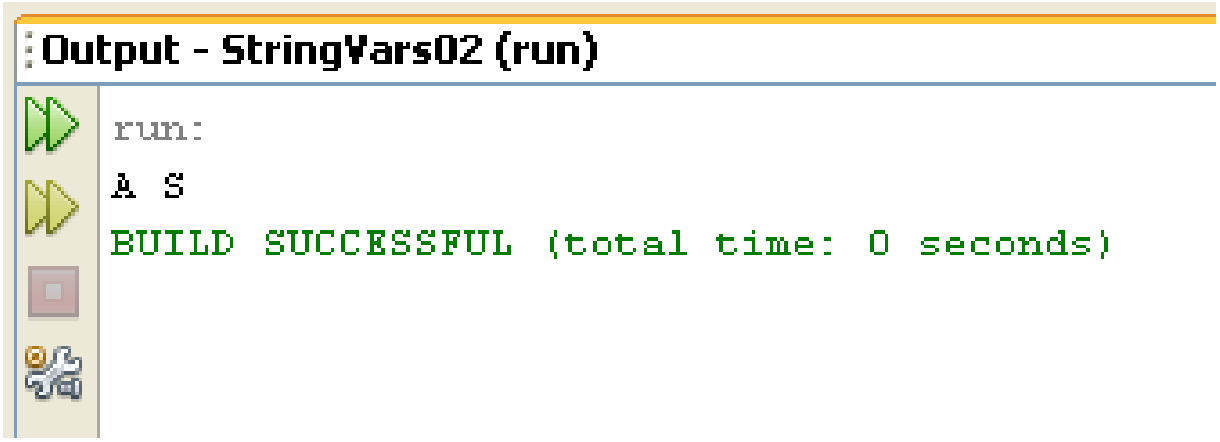
Run your program and you should see this in the Output window:



StringVars02 example

- If you are storing just a single character, then the variable you need is **char** (lowercase “c”).
- To store the character, you use single quotes instead of double quotes.
- Here’s our program again, but this time with the **char variable:**

```
11  public class StringVariables {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          // TODO code application logic here
18          char first_name = 'A';
19          char family_name = 'S';
20          System.out.println(first_name + " " + family_name);
21      }
22  }
23
```


Output - StringVars02 (run)
run:
A S
BUILD SUCCESSFUL (total time: 0 seconds)

Accepting Input from a User

- One of the strengths of Java is the huge libraries of code available to you.
- This is code that has been written to do specific jobs. All you need to do is to reference which library you want to use, and then call a method into action.
- One really useful class that handles input from a user is called the Scanner class.
- The Scanner class can be found in the **java.util library.**
- **To use the Scanner class, you need to** reference it in your code. This is done with the keyword **import.**

import java.util.Scanner;

- The next thing you need to do is to create an object from the Scanner class.
- (A class is just a bunch of code. It doesn't do anything until you create a new object from it.)
- To create a new Scanner object the code is this:

Scanner user_input = new Scanner(System.in);

- So instead of setting up an int variable or a String variable, we're setting up a Scanner variable.
- We've called ours user_input.
- After an equals sign, we have the keyword new.
- This is used to create new objects from a class.

- **The object** we're creating is from the Scanner class.
- In between round brackets we have to tell java that this will be System Input (**System.in**).
- To get the user input, you can call into action one of the many methods available to your new Scanner object.
- One of these methods is called **next.**
- This gets the next string of text that a user types on the keyboard:

```
String first_name;
System.out.print("Enter your first name: ");
first_name = user_input.next( );
```

- Notice that we've used print rather than println like last time.
- The difference between the two is that println will move the cursor to a new line after the output, but print stays on the same line.
- We'll add a prompt for a family name, as well:

```
String family_name;
System.out.print("Enter your family name: ");
family_name = user_input.next( );
```

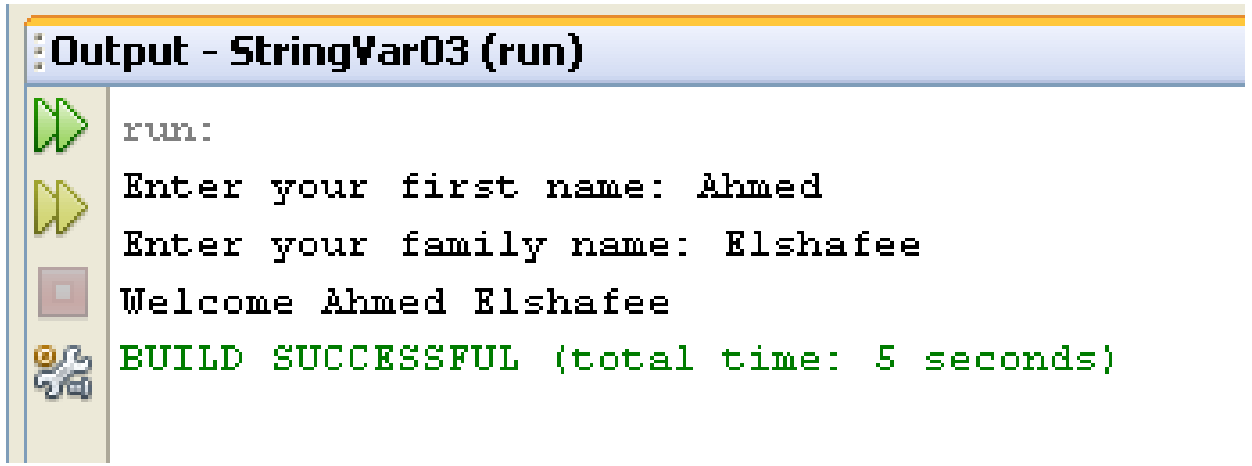
- Then

```
System.out.println("Welcome"+first_name + " " +
family_name);
```

StringVars3

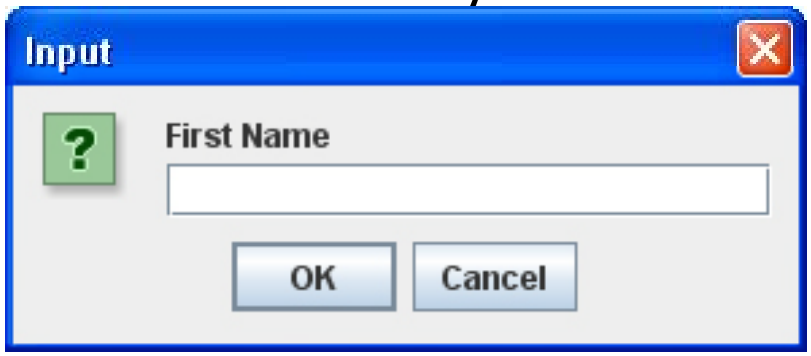
```
package stringvars;

import java.util.Scanner;

public class StringVariables {

    public static void main(String[] args) {

        Scanner user_input = new Scanner(System.in);

        String first_name;
        System.out.print("Enter your first name: ");
        first_name = user_input.next();

        String family_name;
        System.out.print("Enter your family name: ");
        family_name = user_input.next();

        String full_name;
        full_name = first_name + " " + family_name;

        System.out.println("You are " + full_name);

    }
}
```

- Run your program until your Output window displays the following: Java



Option Panes (GUI)

- Another useful class for accepting user input, and displaying results, is the JOptionPane class. This is located in the **javax.swing library.**
- **The JOptionPane** class allows you to have input boxes like this one:



- And message boxes like this:

- The first thing to do is to reference the library we want to use:

import javax.swing.JOptionPane;

- This tells java that we want to use the JOptionPane class, located in the **javax.swing library.**
- In the main function type

String first_name;

first_name = JOptionPane.showInputDialog("First Name");

String family_name;

family_name = JOptionPane.showInputDialog("Family Name");

String full_name;

full_name = "You are " + first_name + " " + family_name;

- To display the result in a message box, add the following:

  JOptionPane.showMessageDialog(null, full_name);

- **Null means** that the message box is not associated with anything else in the program.
- After a comma comes the text we want to display in the message box.
- Notice the line at the bottom of the code:

  System.exit(0);

- As its name suggests, this ensures that the program exits.
- But it also tidies up for us, removing all the created objects from memory.

StringVar04

```
package userinput;
import javax.swing.JOptionPane;

public class InputBoxes {

    public static void main(String[] args) {

      String first_name;
      first_name = JOptionPane.showInputDialog("First Name");

      String family_name;
      family_name = JOptionPane.showInputDialog("Family Name");

      String full_name;
      full_name = "You are " + first_name + " " + family_name;

      JOptionPane.showMessageDialog(null, full_name);
      System.exit(0);

    }
}
```


Input
First Name
Ahmed
OK
Cancel
Input
Family Name
ElShafee
OK
Cancel
Message
You are Ahmed ElShafee
OK

Exercises

Exercise (StringVar05**)**

- Input boxes and Message boxes can be formatted further. Try the following for
- your Input boxes:

showInputDialog("First Name", "Enter Your First Name");

showInputDialog("Family", "Enter Your Family Name");

Exercise (StringVar06**)**

- For your Message boxes try this (yours should be on one line):

showMessageDialog(null, full_name, "Name",
JOptionPane.INFORMATION_MESSAGE);

Exercise

- Instead of JOptionPane.INFORMATION_MESSAGE try these:
- **ERROR_MESSAGE**
- **PLAIN_MESSAGE**
- **QUESTION_MESSAGE**
- **WARNING_MESSAGE**

Exercise (StringVar07**)**

- Input boxes are not just used for text: they can accept numbers as well.
- Write a program that prompts the user for two numbers, the breadth of a rectangle and the height of a rectangle.
- Use a message box to calculate the area of the rectangle. (Remember: the area of a rectangle is its breadth multiplied by the height.)
- However, you'll need some extra help for this exercise.

Help for the Exercise

- You have to use the String variable to get your numbers from the user:

String breadth;

breadth = JOptionPane.showInputDialog("Rectangle Breadth");

- However, you can't multiply two strings together.
- You need to convert the Strings to integers.
- You can convert a string to an integer like this:

Integer.parseInt(string_to_convert)

- You can then multiply and assign on the same line;

  int area = Integer.parseInt(string_one) * Integer.parseInt(string_two);

- For the message box, use concatenation:

  “answer = ” + area

- You can use any of the MESSAGE symbols for your message

Exercise (StringVar08**)**

- The programme will crash if you enter floating point values for the breadth and height.
- How would you solve this?
- When you have solved the above exercise, do you really want Integer.parseInt?
- What else do you think you can use?

Thanks,

See you next Lecture, isA