Creating a Java Program

Objectives

After completing this lesson, you should be able to:

- Create a Java class
- Write a main method
- Use `System.out.println` to write a String literal to system output

Java Classes

A Java class is the building block of a Java application.

Includes code that:

- Allows a customer to add items to the shopping cart
- Provides visual confirmation to the customer

Program Structure

- A class consists of:
  - The class name. Class names begin with a capital letter.
  - The body of the class surrounded with braces { }
    - Data (called fields)
    - Operations (called methods)
- Example:

Java is case-sensitive!

```
public class Hello {
   // fields of the class
   // methods
}
```

Java Packages

- A package provides a namespace for the class.
  - This is a folder in which the class will be saved.
  - The folder name (the package) is used to uniquely identify the class.
  - Package names begin with a lowercase letter.
- Example:

Package name

```
package greeting;

public class Hello {
    // fields and methods here
}
```

The class's unique name is: greeting.Hello

The `main` Method

- It is a special method that the JVM recognizes as the starting point for every Java program.
- The syntax is always the same:

```
public static void main (String args[])  {
   // code goes here in the code  block
}
```

- It surrounds entire method body with braces { } .

A `main` Class Example

Class name

```
public class Hello {

  public static void main (String[] args)  {
    // Entry point to the program.
    // Write code here:
    System.out.println ("Hello World!");
  }
}
```

Comments

`main` method

Program output

Output to the Console

- Syntax:

```
System.out.println (<some string value>);
```

- Example:

String literal

```
System.out.println ("This is my message.");
```

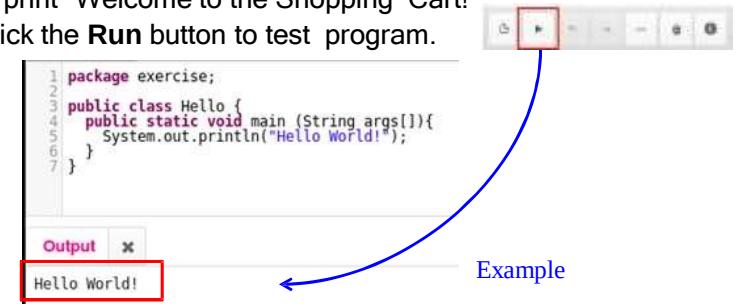
Fixing Syntax Errors

- If you have made a syntax error, the error message appears in the Output panel.
- Common errors:
  - Unrecognized word (check for case-sensitivity error)
  - Missing semicolon
  - Missing close quotation mark
  - Unmatched brace

Exercise 3-2: Creating a `main` Method

In this practice, you manually enter a `main` method that prints a message to the console.

- In the code editor, add the `main` method structure to the ShoppingCart class.
- In the code block, use a `System.out.println` method to print “Welcome to the Shopping Cart!”
- Click the **Run** button to test program.



Example

Quiz

Which `main` method syntax is correct?

a. `Public static void main (String[ ] args){ }`
b. `public Static void Main (String[ ] args){ }`
c. `public static void main (String ( ) args)[ ]`
d. `public static void main (String[ ] args){ }`

Summary

In this lesson, you should have learned how to:

- Create a class using the Java Code Console
- Create (declare) a Java class
- Define a main method within a class
- Use `System.out.println` to write to the program output
- Run a program in the Java Code Console

Data in a Cart

Objectives

After completing this lesson, you should be able to:

- Describe the purpose of a variable in the Java language
- List and describe four data types
- Declare and initialize `String` variables
- Concatenate `String` variables with the '+' operator
- Make variable assignments
- Declare and initialize `int` and `double` variables
- Modify variable values by using numeric operators
- Override default operator precedence using ( )

Topics

- Introducing variables
- Working with `String` variables
- Working with numbers
- Manipulating numeric data

Variables

- A variable refers to something that can change.
  - Variables can be initiated with a value.
  - The value can be changed.
  - A variable holds a specific type of data.

The **type** of data

Variable name

The **value** of the variable

```
String firstName = "Mary";
```

Variable Types

- Some of the types of values a variable can hold:
  - `String` (example: "Hello")
  - `int` (examples: -10, 0, 2, 10000)
  - `double` (examples: 2.00, 99.99, -2042.09)
  - `boolean` (true or false)
- If uninitialized, variables have a default value:
  - `String` : "" (the empty string)
  - `int`: 0
  - `double`: 0.0
  - `boolean`: false

Naming a Variable

Guidelines:

- Begin each variable with a lowercase letter. Subsequent words should be capitalized:
  - `myVariable`
- Names are case-sensitive.
- Names cannot include white space.
- Choose names that are mnemonic and that indicate to the casual observer the intent of the variable.
  - `outOfStock (a boolean)`
  - `itemDescription (a String)`

Uses of Variables

- Holding data used within a method:

```
String name = "Sam" ;
double price = 12.35;
boolean outOfStock = true;
```

- Assigning the value of one variable to another:

```
String name = name1;
```

- Representing values within a mathematical expression:

```
total = quantity * price ;
```

- Printing the values to the screen:

```
System.out.println(name);
```

Topics

- Introducing variables
- Working with `String` variables
- Working with numbers
- Manipulating numeric data

Variable Declaration and Initialization

- Syntax :

```
type identifier [=value];
```

- Examples:

```
String customer;

String name, city;

String address = "123 Oak St";

String country = "USA", state = "CO";
```

Variable declared

Two variables declared

Variable declared and initialized

Two variables declared and initialized

`String` Concatenation

- `String` variables can be combined using the '+' operator.

```
                    - + stringVariable2
stringVariable
                    1
                    - + "String literal"
stringVariable
```

- Example:

```
String greet1 = "Hello";
String greet2 = "World";
String message = greet1 + " " + greet2 + "!";
String message = greet1 + " " + greet2 + " " + 2014 +"!";
```

le2

`String` Concatenation Output

You can concatenate `String` variables within a method call:

```
System.out.println(message);
System.out.println(greet1 + " " + greet2 +  "!");
```

Output:

```
Hello World!
Hello World!
```

Exercise 4-1: Using `String` Variables

In this exercise, you declare, initialize, and concatenate `String` variables and literals.

Topics

- Introducing variables
- Working with `String` variables
- Working with numbers
- Manipulating numeric data

`int` and `double` Values

- `int` variables hold whole number values between:
  - `-2,147,483,648`
  - `2,147,483,647`
  - Examples: 2, 1343387, 1_343_387
- `double` variables hold larger values containing decimal portions.
  - Use when greater accuracy is needed.
  - Examples: 987640059602230.7645 , -1111, 2.1E12

Initializing and Assigning Numeric Values

- `int` variables:
  - `int quantity = 10;`
  - `int quantity = 5.5;`

Compilation fails!

- `double` variables:
  - `double price = 25.99;`
  - `double price = 75;`

Run time will interpret as 75.0.

Topics

- Introducing variables
- Working with `String` variables
- Working with numbers
- Manipulating numeric data

Standard Mathematical Operators

Purpose	Operator	Example	Comments
Addition	`+`	`sum = num1 + num2;` If `num1` is 10 and `num2` is 2, `sum` is 12.	
Subtraction	`-`	`diff = num1 - num2;` If `num1` is 10 and `num2` is 2, `diff` is 8.	
Multiplication	`*`	`prod = num1 * num2;` If `num1` is 10 and `num2` is 2, `prod` is 20.	
Division	`/`	`quot = num1 / num2;` If `num1` is 31 and `num2` is 6, `quot` is 5.	Division by 0 returns an error. The remainder portion is discarded.

Increment and Decrement Operators (++ and --)

The long way:

```
age = age + 1;
```

or

```
count = count - 1;
```

The short way:

```
age++;
```

or

```
count--;
```

Operator Precedence

Here is an example of the need for rules of precedence.

Is the answer to the following problem 34 or 9?

```
int c = 25 - 5 * 4 / 2 - 10 +  4;
```

Operator Precedence

Rules of precedence:

1. Operators within a pair of parentheses
2. Increment and decrement operators (++ or --)
3. Multiplication and division operators, evaluated from left to right
4. Addition and subtraction operators, evaluated from left to right

Using Parentheses

Examples:

```
int c = (((25 - 5) * 4) / (2 - 10)) + 4;
int c = ((20 * 4) / (2 - 10)) + 4;
int c = (80 / (2 - 10)) + 4;
int c = (80 / -8) + 4;
int c = -10 + 4;
int c = -6;
```

Summary

In this lesson, you should have learned how to:

- Describe the purpose of a variable in the Java language
- List and describe four data types
- Declare and initialize `String` variables
- Concatenate `String` variables with the '+' operator
- Make variable assignments
- Declare and initialize `int` and `double` variables
- Modify numeric values by using operators
- Override default operator precedence using ( )