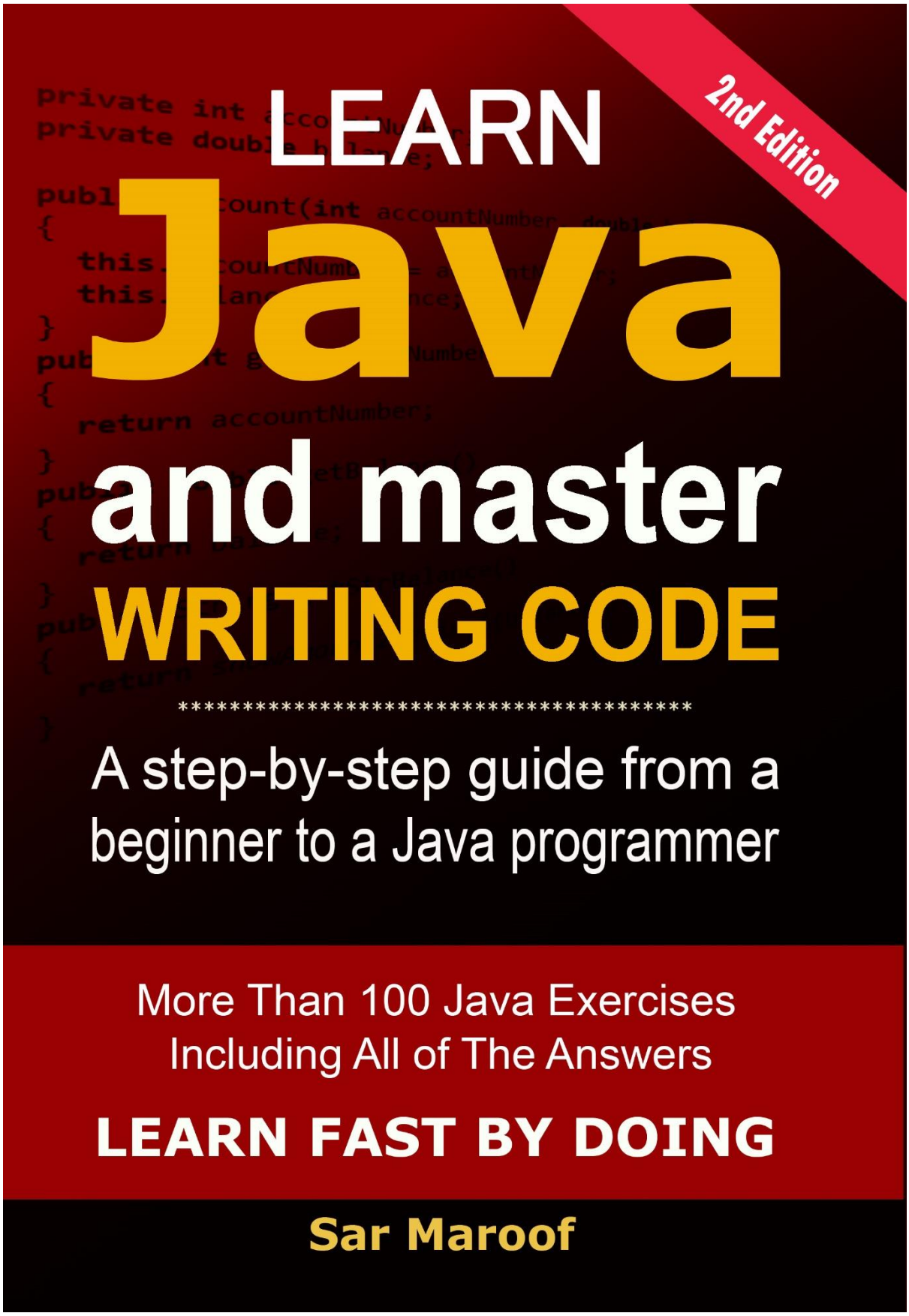

2nd Edition
LEARN
Java
and master
WRITING CODE
****
A step-by-step guide from a beginner to a Java programmer
More Than 100 Java Exercises
Including All of The Answers
LEARN FAST BY DOING
Sar Maroof

Download the source code: **www.SarMaroof.com**

Learn Java and Master Writing Code

- A step-by-step guide from a beginner to a Java programmer
- More than 100 Java code exercises, including all of the answers at the end of this book.
- Download the source code and learn fast by practicing and doing
- Learn to read Java code, understand it, change it and test it to see the results.
- Write your own Java-code.
- Learn the basic concept of object-oriented programming
- Prior experience in coding is not required in order to start this book

Sar Maroof

Copyright © 2017 Sar Maroof

All rights reserved.

Table of Contents

Introduction

This book is organized to teach Java to beginners, and it guides readers to master writing code by working with more than 100 examples, exercises, and assignments.

Prior experience in coding is not required in order to start this book. It explains Java in an easy to understand way, with simple coding examples and many exercises that make it interesting and helpful to the reader even if they have no prior experience in programming.

It is difficult for any expert software developer to believe that anyone can learn to program by only reading books. All experts build their experiences by doing and practicing programming. That is the exact reason why this book focuses also on working with a complete code that is specially designed for anyone who wants to learn Java.

You can download the source code for this book on the website of the author, **www.sarmaroof.com**, and follow the guide to set up the code in a Java development environment.

Once you establish the code, you can start to enjoy working with it and writing your own code by executing the exercises and the tasks of the book. By clicking on one button, you can compile and run each code and see the effect that your code has.

Sar Maroof is a professional software development teacher. He publishes technical articles and has also worked for more than a decade in developing software for large, as well as small companies, and also as a freelancer.

Sar Maroof

Is This Book for You?

If you ask the question whether this book is for you, the answer is to look at the following simple Java code, which is a simple class that describes cars. If you don't understand a single line of the following code, you are a real beginner. Don' t worry because this book assumes that you are a beginner. It starts at the very beginning of understanding programming. By following the chapters you will gradually learn standard Java, understand it and start writing your own code.

If you understand the following code completely and you have some experiences with programming, this book will help you to improve your Java skills.

I would recommend beginners to start from the beginning and read the "Basic Knowledge".

Example 1

```
class Car
{
  String brand;
  int mileage;
  double price;

  public static void main(String[] args)
  {
    Car car1 = new Car();
    car1.brand = "Nissan";
    car1.mileage = 7000;
    car1.price = 6500.25;

    System.out.print( car1.brand + ", ");
    System.out.print( car1.price + ", ");
    System.out.print( car1.mileage);
  }
}
```

Basic Knowledge

This book focuses on learning programming by reading the explanation and practicing with programming code. That is far more interesting and easier to learn than only reading the theoretical explanation. I would strongly recommend to download the source code of this book on my website and follow the step by step explanation of how to install JSE (Java Standard Edition). Notice that some versions of Java are called JDK (Java Development Kit). You can find on the previously mentioned website the explanation of how to set up the source code of this book in an IDE (Integrated Development Environment). Start to read the following Basic Knowledge so that you understand how programming works and what kinds of tools and software you need to compile and run a Java code.

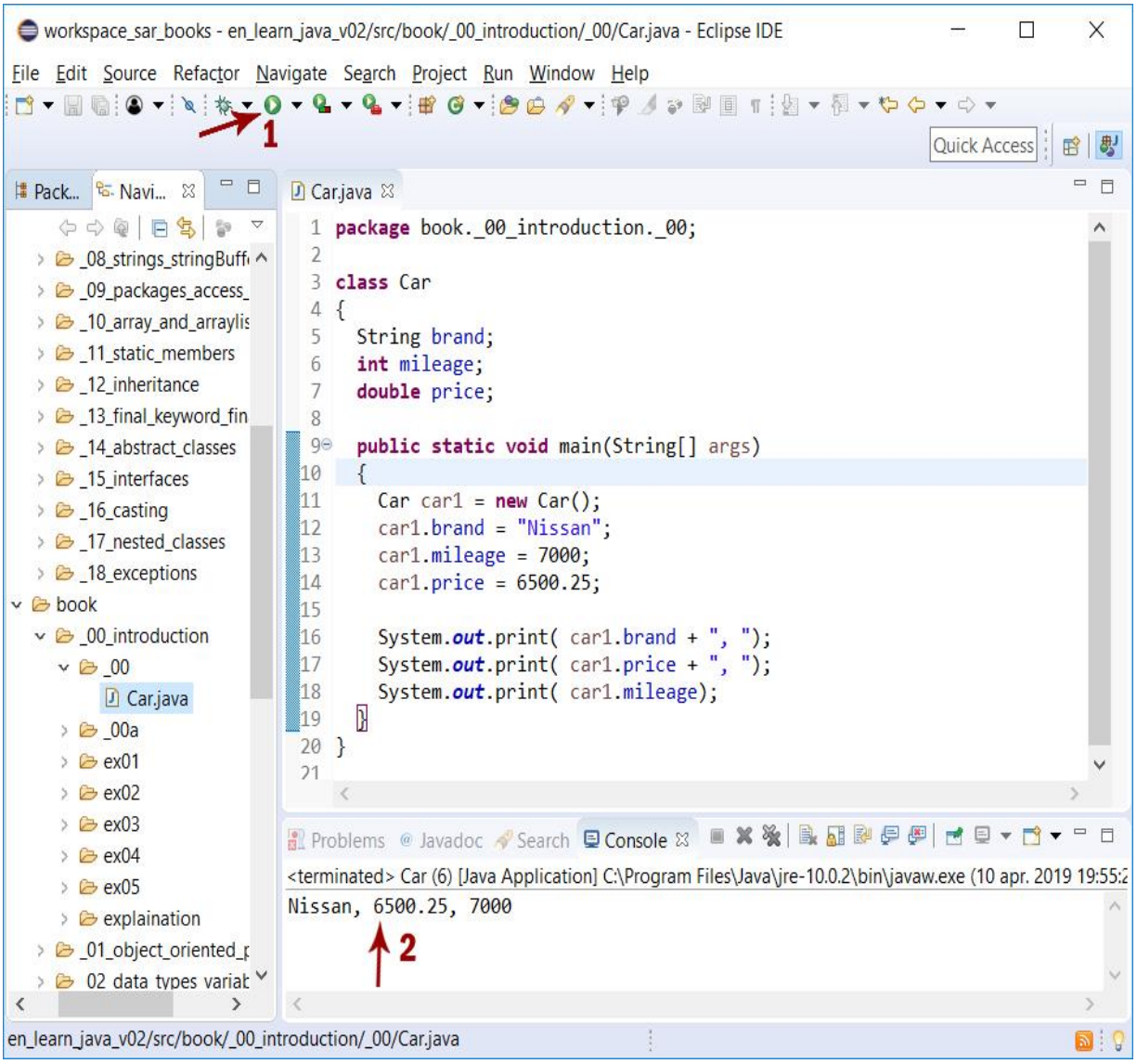
1. Java Editor Ide (Integrated Development Environment)

We use as a Java editor, Eclipse including JSE (Java Standard Editon), which are both free to download. Java is a platform-independent programming language, which is why you can run Java programs on every operating system.

2. Compiling And Running Programs

Many beginners might scare to hear "compiling and running a program", but in fact, you can do that by clicking on the green button, see arrow one in the picture below. By compiling and running the following program "Nissan, 6500.25, 7000" is written to the standard output, see arrow two in the picture below. Compiling is the translation of source code into machine language with the help of a tool called a compiler. After compiling a source code, the program is directly executable.

The picture below shows how Eclipse IDE looks like, which we use to edit Java programs. The arrow two points to the standard output of the program. When you compile and run the program, you see the output in that part of the screen.


workspace_sar_books - en_learn_java_v02/src/book/_00_introduction/_00/Car.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
1
Quick Access
Pack... Navi...
_08_strings_stringBuff
_09_packages_access_
_10_array_and_arraylis
_11_static_members
_12_inheritance
_13_final_keyword_fin
_14_abstract_classes
_15_interfaces
_16_casting
_17_nested_classes
_18_exceptions
book
_00_introduction
_00
Car.java
_00a
ex01
ex02
ex03
ex04
ex05
explaination
_01_object_oriented_
02 data types variat
Car.java
1 package book._00_introduction._00;
2
3 class Car
4 {
5 String brand;
6 int mileage;
7 double price;
8
9 public static void main(String[] args)
10 {
11 Car car1 = new Car();
12 car1.brand = "Nissan";
13 car1.mileage = 7000;
14 car1.price = 6500.25;
15
16 System.out.print(car1.brand + ", ");
17 System.out.print(car1.price + ", ");
18 System.out.print(car1.mileage);
19 }
20 }
21
Problems Javadoc Search Console
<terminated> Car (6) [Java Application] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (10 apr. 2019 19:55:2
Nissan, 6500.25, 7000
2
en_learn_java_v02/src/book/_00_introduction/_00/Car.java

3. Java Classes And Interfaces

Java programs are mostly classes and interfaces. These two concepts are covered in details later in this book. For now, you need only to learn that a class in Java begins with a statement as **class Person**. Person is the name of the class, that can be chosen by programmers. Every class in Java is stored in a file with the name of the class and the extension of **.java**. The class Person should be stored in a file called **Person.java**. Each class name begins with the keyword class, and each interface name begins with the keyword interface. Classes and interfaces also have members like variables and methods. Methods are blocks, which are written between curly braces.

Example 2
The following is a class that describes products and is stored in a file with the name Product.java

```
class Product
{
  // code
}
```

The following interface must also be stored in a file with the name Payable.java.

Example 3

```
interface Payable
{
  // code
}
```

4. Statements

Statements in Java are similar to sentences in natural languages and are executable units. A statement usually ends with a semicolon (;).

Examples of statements are:

```
String name = "Emily";
int age = 22;
```

5. Code Block

The code in Java is within a start and an end brace, and it is called a block of code. Below are some examples of block codes.

Block type	Block style

Class	`class Computer`
	`{`
	`  // code`
	`}`

Method	`void setBrand()`
	`{`
	`  // code`
	`}`
	The keyword void means that the method doesn't return any values.

Conditional statements	`if(age < 18)`
	`{`
	`  discount = %10;`
	`}`

Iteration	`for( int i=0; i<5; i ++)`
	`{`
	`  // code`
	`}`

6. The Main Method

The main method is a method that is needed to execute a Java program. The main method begins with **public static void main(String[] args)** and is followed by a block code as shown below.

```
public static void main(String[] args)
{
    // code
}
```

For the moment it is important to learn the following about the main method.

- All executable programs must have a main method such as the code below.
- The statements within the block of the main method are executed.
- The execution of the statements inside the main method block is done from top to bottom.
- In the next example, **statement 1** is first executed, then **statement 2,** and at last
- **statement 3**.
- The following code writes **253** to the standard output.

Example 4

```
class Calculator
{
  // statements;
  // main method
  public static void main(String[] args)
  {
    System.out.print(2); // statement 1;
    System.out.print(5); // statement 2;
    System.out.print(3); // statement 3;
  }
}
```

7. Writing Values Of Variables And Texts To The Standard Output

The statement **System.out.println();** is used to write values of variables and texts to the standard output and it is a practical way to test the output of your program. It is not necessary for now to understand all the details about this statement, but it is important to know that you can use it to test the output of your programs.

If you compile and run the next code, the following is written to the standard output:

Ronald

35

My name is Emma.

Age: 35 year

Example 5

```
class Doctor
{
  public static void main(String[] args)
  {
    String name = "Ronald";  // variable name is a String type
    int age = 35; // Variable age is an integer type

    // statement 1 writes the value of name
    // to the standard output
    System.out.println(name);
    // statement 2 writes the value of age
    // to the standard output
    System.out.println(age);
    // statement 3 writes the text "My name is Emma."
    // to the standard output
    System.out.println("My name is Emma.");
    /*
     *  statement 4 writes a combination of a text and
     *  a value of a variable to the standard output
     */
    System.out.println("Age: " + age + " year");
```

```
  }
}
```

To write a text to the standard output, it must be between quotation marks, but for the values of variables that are not required, see the **statements 1, 2 and 3** in the previous example.
To write a combination of texts and values of variables to the standard output, you need to use the plus (+) sign between the variable name and the text, see **statement 4** in the previous example. To write the texts and the variables on one line to the standard output you need to use
System.out.print instead of **System.out.println.**

8. Comments

Comments are ignored by the compiler. Below are two ways to add comments to your code.

1. Comment of a single line begins with two slashes //.
   Everything on the right side of these characters, see Java as comments:

```
// comment of a single-line
```

2. Comment of multiple lines starts with /* and ends with */.
   Everything between these two characters sees Java as comments:

```
/*
  here is a comment
  of multiple lines
*/
```

9. The Keyword Public

Java supports controlling access to the classes and class members by using special keywords. Using the **public** keyword before the keyword **class** or the class members (variables and methods) indicates that the class or the member is accessible from other classes. This will be covered in this book later in details.

Important to Know

1 Java Standard API (Application Programming Interface)

Java provides a lot of code that can be reused by programmers. It is important for every Java programmer to use this free rich library, which is why it is introduced in this book. For some of the assignments, you need to open this document which helps with how to use the code. You can find the Java standard API documentation on the website of Oracle.

2 Escape Sequences

An escape sequence is a character preceded by a backslash. These characters have a special meaning for the compiler. For example, if you have a double quote within a double quote, you must use escape sequences,

see the following example. If you need to write the text "He says: "I go to London."" to the standard output, you should do that as follows.

```
System.out.print("He says: \"I go to London\".");
```

Example 6

```
class Symbol
{
  public static void main(String[] args)
  {
    System.out.println("Apostrophe     : " + "abcde\'fghij");
    System.out.println("Quotation mark : " + "abcde\"fghij");
    System.out.println("Backslash      : " + "abcde\\fghij");
    System.out.println("New line       : " + "abcde\nfghij");
    System.out.println("New line 2     : " + "abcde\ffghij");
    System.out.println("Tab            : " + "abcde\tfghij");
    System.out.print("It was written \"Parking is not Allowed.\".");
  }
}
```

If you compile and run the above code, the following is written to the standard output.

```
Apostrophe     : abcde'fghij
Quotation mark : abcde"fghij
Backslash      : abcde\fghij
New line       : abcde
fghij
New line 2     : abcde_fghij
Tab            : abcde    fghij
```

It was written "Parking is not Allowed.".

The table of the escape sequences:

Escape sequence	Description
\'	single quote
\"	double quote
\\	backslash character
\b	backspace
\n	Newline
\r	carriage return
\t	Tab
\f	form feed

1. Object Oriented Programming

Java is an object-oriented programming language. The OO techniques allow you to build computer programs using objects that interact and communicate with each other.
Each object contains variables and methods through which they work together. Java sees everything as an object so, people, animals, plants, and things around you are treated as objects.
Example of objects are: **student, employee, customer, company, chair, laptop, mobile, animal, plant** and so on.

To create an object, you need first to create a template or a description of the object, and that is called a **class**. Below is explained how to create a Java class. Once you create a class you can instantiate unlimited numbers of objects from it.

To make it clear how Java deals with classes, we use the following example. Suppose you are asked to write a small program for an employer who saves the names, ages, and salaries of his employees. The name, age, and salary are called the **attributes** or the **variables** of the class.

To write this simple program, we have to take the next steps.

1.1. Creating A Class Employee

- As has already been explained in this book, each Java class is stored in a file with the name of the class and the extension **.Java**. In this case, we store the class Employee in the file **Employee.Java**.
- A class starts with the Java keyword class followed by the name of the class and a block of code as shown in the next example:

The employer wants to keep the **names, ages,** and **salari**es of his employees. Within the class **Employee**, we declare variables for these three attributes.

Each attribute starts with a data type or variable type (more details about data types will be covered later in this book. In this example, we use the following three data types.

Data types	Used for
String	The string type is used in Java for a combination of symbols, numbers and letters such as names and texts.
int	The int type is used for integers such as ages.
double	The double type is used for decimal numbers such as prices and salaries.

Example 1

```
class Employee
{
  // the attributes (variables) of the class
```

```
  String name;
  int age;
  double salary;

  // the main method to execute the code
  public static void main(String[] args)
  {
    // more code
  }
}
```

So far we have created the class Employee, we can create as many employee objects as we want.

1.2. Creating Objects Based On The Class Employee

- In order to execute the Employee class, the main method should be defined as it is already explained in this book.
  - We can create as many employee objects as we need. As examples, we make two objects for the employees Emma and Jack.
  - An object has a name and you may choose any name you wish for the objects you create.
  - In the following example, we create the object **employee1** for Emma and the object **em-ployee2** for Jack. Remember that Emma and Jack are the names of the two employee objects. The object in our example contains the names, the ages and the salaries of Emma and Jack.
  - In Java we can create an object by first writing the name of the class then the name of the object follows by the equal sign and the Java keyword **new** then the name of the class and ends by open and closed parentheses. At last, a semicolon as shown below.

```
// create object employee1
Employee employee1 = new Employee();
```

Notice: Java class names start always with capital letters, while the names of objects and attributes start with small letters.

Example 2

```
class Employee
{
  // the attributes (variables) of the class
  String name;
  int age;
  double salary;

  // the main method to execute the code
  public static void main(String[] args)
  {
    // create object employee1
```

```
    Employee employee1 = new Employee();
    // create object employee2
    Employee employee2 = new Employee();
  }
}
```

1.3. Assigning Values To The Variables

To assign values to the variables, we use the name of the object separated from the variable name by a dot (.) then the equal sign and at last the value of the variable.
The following statements assign the values "Emma, 24, 2500.55" respectively to the variables name, age, and salary of the object employee1.

Note: the value of a string should be between quotes as shown below.

```
employee1.name = "Emma";
employee1.age = 24;
employee1.salary = 2500.55;
```

The following code doesn' t contain errors, but nothing will be visible if you compile and run it. The reason is that we only told the computer to memorize the names, ages, and salaries of two employee objects namely Emma and Jack. That is comparable with human' s memory when you memorize the multiplication tables, for example, you know that 6 x 5 is equal to 30, but you give the answer only when someone asks you or you need it.

Example 3

```
class Employee
{
  String name;
  int age;
  double salary;

  public static void main(String[] args)
  {
    // create object employee1
    Employee employee1 = new Employee();
    /*
     *  assign values to the
     *  variables of the object employee1
     */
    employee1.name = "Emma";
    employee1.age = 24;
    employee1.salary = 2500.55;

    // create object employee2
    Employee employee2 = new Employee();
    /* assign values to the
     * variables of the object employee2
     */
    employee2.name = "Jack";
    employee2.age = 32;
    employee2.salary = 2800.45;
```

```
  }
}
```

1.4. Accessing The Attribute's Values Of The Objects

To access the variables of the objects, you can use the name of the object and the variable name separated by a dot "." as shown below. In this step, we try to ask the program to write the data of the objects " employee1 and employee2" to the standard output. It is obvious that everyone can memorize the data of two employees of our example, but the power of the computer is that it can memorize almost unlimited data of unlimited objects and that is beyond the capability of humans. In the following examples, we ask the program to show the data that we programmed as explained before in this book by using the following statement.

```
System.out.print(employee1.name + ", ");
```

If the next program is compiled and run, the following is written to the standard output.

```
Emma, 24, 2400.55
Jack, 32, 2800.45
```

Example 4

```
class Employee
{
  String name;
  int age;
  double salary;

  public static void main(String[] args)
  {
    // create object employee1
    Employee employee1 = new Employee();
    /* assign values to the
     * variables of the object employee1
     */
    employee1.name = "Emma";
    employee1.age = 24;
    employee1.salary = 2400.55;

    // create object employee2
    Employee employee2 = new Employee();
    /* assign values to the
     * variables of the object employee2
     */
    employee2.name = "Jack";
    employee2.age = 32;
    employee2.salary = 2800.45;
```

```
        // print employee data;
        System.out.print(employee1.name + ", ");
        System.out.print(employee1.age+ ", " );
        System.out.println(employee1.salary);
        System.out.print(employee2.name + ", ");
        System.out.print(employee2.age+ ", " );
        System.out.println(employee2.salary);
    }
}
```

There are some important principles of object-oriented programming such as Inheritance, and Encapsulation. In the following chapters, you will find more explanation of object-oriented programming using Java code. Remember that this book is about learning Java as a programming language, but offers only some basic principles of object-oriented programming.

1.5. Inheritance

Inheritance is one of the important principles of object-oriented programming. The concept of Inheritance supports reusability of code that has already been written. To reuse the existing code you create parent and child classes. The parent classes are called super classes and a child class, which extends the code of the superclass is called a subclass. You can find more details about inheritance and how it works with code examples in the chapter Inheritance.

1.6. Polymorphism

This is the possibility of using a single method name for multiple methods by applying specific rules. There are two types of polymorphism, which are explained below.

- **Overriding methods**

By applying this technique you can override the superclass methods in the subclass. The subclass method has the same name, the same type and the same number of arguments as the superclass.

- **Overloading methods**

In the case of overloading, the methods also have the same name, but they have different types and/or different number of arguments.

You can find more details about overriding and overloading methods in the chapter Inheritance.

1.7. Encapsulation

This concerns the possibility of concealing data by limiting the direct access to the variables, but instead of that using the public accessors and mutators. Accessories are used to provide information

about the status of an object, while you can use mutators for changing the status of an object. Accessors in Java starts with the word get, while mutators start with the word set. They are also called getters and setters. In the chapter Methods, Package and access modifiers, you will learn more about methods and accesses.

Quiz 1: Objects of the Class Car

What happens when the following program is compiled and run?

```
class Car
{
  String brand;
  int mileage;
  double price;

  public static void main(String[] args)
  {
    Car car1 = new Car();
    Car car2 = new Car();
    Car car3 = new Car();

    car1.brand = "Volkswagen";
    car1.mileage = 4000;
    car1.price = 4500.75;
    car2.brand = "Mazda";
    car2.mileage = 2000;
    car2.price = 3500.65;
    car3.brand = "Nissan";
    car3.mileage = 7000;
    car3.price = 6500.25;

    System.out.print(car2.brand + ", ");
    System.out.print(car3.price + ", ");
    System.out.print(car1.mileage);
  }
}
```

Select the correct answer.

a) This code writes Volkswagen, 4500.75, 4000 to standard output.
b) This code writes Mazda, 3500.65, 2000 to standard output.
c) This code writes Mazda, 6500.25, 4000 to standard output.
d) This code writes Volkswagen, 3500.65, 4000 to standard output.
e) This code writes Nissan, 3500.65, 4000 to the standard output.

Explanation

car2.brand writes Mazda to the standard output, because the statement car2.brand = "Mazda"; assigns the value Mazda to the variable brand of car2.
car3.price writes 6500.25 to the standard output, because the statement car3.price = 6500.25; assigns the value 6500.25 to the variable price.
car1.mileage writes 4000 to the standard output, because the statement
car1.mileage = 4000; assigns the value of 4000 to the variable mileage of car 1.

The correct answer is: c.

Exercises

For all of the car objects, you want to add the color of the car. Assume that the color of the object car1 is black, of car2 is red and car3 is white.

Update the class Car so that the following should be written to the standard output if the code is compiled and run.

Brand 1: Volkswagen, color: black

Brand 2: Mazda, color: red

Brand 3: Nissan, color: white

Quiz 2: Objects of the Class Animal

What happens when the following code is compiled and run?

```
class Animal
{
  String name;
  String sort = "pet";

  public static void main(String[] args)
  {
    Animal animal1 = new Animal();
    Animal animal2 = new Animal();
    Animal animal3 = new Animal();

    animal1.name = "Tiger";
    animal1.sort = "predator";
    animal2.name = "Dog";
    animal3.name = "cow";

    System.out.print(animal1.name +", "+animal3.sort);
  }
}
```

Select the correct answer.

a) This code writes Tiger, pet to the standard output.
b) This code writes Tiger, predator to the standard export.
c) This code writes cow, predator to the standard export.
d) This code writes Dog, pet to the standard output.

Explanation

The first part of the statement System.out.print (animal1.name +", "+animal3.sort); is "animal1.name". The statement animal1.name = "Tiger"; assigns the value Tiger to the name of the object animal1.

The initial value of sort is pet. If there is no value assigned with the variable sort the initial value remains the value of the variable sort, which is pet. That is why the second part of "animal3.sort" writes the sort pet for the animal3 object to the standard output.

The correct answer is: a.

Exercises

The above code does not work as desired. Improve the code so that the following is written to the standard output by compiling and running the code.
Animal 1: Tiger, Sort: predator
Animal 2: Dog, Sort: pet
Animal 3: cow, Sort: farm animal

Quiz 3: The Number of Objects of the Class Computer

How many computer objects are created in the next program?

```
class Computer
{
  String brand;
  int hardDisk;
  int ram;

  public static void main(String[] args)
  {
    Computer comp = new Computer();
    Computer myComputer = new Computer();
    Computer aComputer = new Computer();
    Computer computer = new Computer();
  }
}
```

Select the correct answer.

a) This code does not create any computer objects.
b) This code creates three computer objects.
c) This code creates four computer objects.
d) This code creates five computer objects.
e) This code creates two computer objects.

Explanation

There are four statements in this program that create objects using the Java keyword new. Those statements create the objects comp, myComputer, aComputer and computer. So, there are four computer objects created.

The correct answer is: c.

Exercises

Assign the values HP, 500 and 8 respectively to the variables brand, hard disk, and ram of the object myComputer. By compiling and running the program, the following should be written to the standard output.

Brand: HP
Hard disk: 120 GB
RAM: 8 GB

2. Data Types & Variables

2.1 What are variables?

A variable is a memory location, which could be used to store a value. Variables have names that are chosen by programmers. A variable should be declared before using it and its value can be changed.

There are 8 primitive data types (variables) in Java. Those types are: **byte, short, int, long, float, double, char and boolean,** which are divided into 4 categories as shown below.

Data Type (bits)	**Range, Description and Examples**
Integer Type	
byte (8 bits)	-2^7 to $2^7 - 1$, -128 to 127 The byte type is small and can be used to save memory. Its default value is 0. **Example: byte b = 20;**
short (16 bits)	-2^{15} to $2^{15} - 1$ The short type can also be used to save memory. Its default value is 0. **Example: short s = 500;**
int (32 bits)	-2^{31} to $2^{31} - 1$ The int type can be used for bigger values. Its default value is 0. **Example: int i = 2500;**
long (64 bits)	-2^{63} to $2^{63} - 1$ The long type can be used when you need a range of values wider than those of int. Its default value is 0. **Example: long l = 23333333333;**
Floating-point Type	
float (32)	~ -3.4×10^{38} to ~ 3.4×10^{38} The float type can be used when floating-point types and values are needed. **Example: float f = 1.4f**
double (64)	~ -1.8×10^{308} to ~1.8×10^{308} The double type can be used when floating-point types and values are needed. double is a default choice for decimal values. **Example: double d = 22.3;**
Character Type	
char (16)	0 to 65,535 The char type can be used by character types like a, b, c, d, $ Characters of type char are always inside single quotes. For the type char you can use a Unicode character as 'B' or as a hexadecimal number of '\u0000' to '\uFFFF'.

	Examples: '\u03A9' = Ω '\u0045' = E '\u20AC' = € **Example: char letter = 'd';**
Boolean Type	
boolean (1)	The boolean type has two possible values either **true** or **false**. It is false by default. **Example: boolean bool = true;**
String	A string is used for the texts in Java, it is not a primary variable, but it is an object. This book covers string in a separate chapter. Texts of the type string are between double quotes. **Example: String text = "My name is John";**

Data Type	Default Value
byte, short, int, long	0
float, double	0.0
char	'\u0000' (the null character)
boolean	false
non-primitive data types (object)	null Objects are explained later in this book.

2.2 Declaring And Initializing Variables

Variable type (always required)
Identifier (always required)
Initial value (not always required)

Example 1

```
double price = 44.00;
int height;
```

The variable height has the default value of 0, because it is not initialized.
There are three types of variables in Java, namely local variables, instance variables and class variables.

Instance variables: An instance variable is declared within a class, but outside of the methods, constructors or other blocks.
Local variables: A local variable is a variable that is declared within a method, constructor or a block.
Class variables: Class variables are also called static variables. They are declared once inside a class but outside the methods, constructors or blocks. There is only one copy of the class variable is available.

Example 2

```
class MyClass
{
  double wage;          // instance variable
  static int counter;   // class variable

  void myMethod()       // method
  {
    char gender = 'm'; // local variable
  }
}
```

Quiz 1: Primitive data types and variables

What happens when the following program is compiled and run?

```
class Worker
{
  boolean isMarried;
  int age = 29;
  long bankAccount = 6552;
  double wage = 110.30;
  char gender = 'm'; // female: f, male: m

  public static void main(String[] args)
  {
    Worker wk = new Worker();

    System.out.print(wk.age + ", ");
    System.out.print(wk.bankAccount + ", ");
    System.out.print(wk.wage + ", ");
    System.out.print(wk.isMarried + ", ");
    System.out.print(wk.gender);
  }
}
```

Select the correct answer:

a) This code writes "29, 6552, 110.3, false, m" to the standard output.
b) This code writes "29, 6552, 110.3, true, m" to the standard output.

Explanation

All the values of the variables are printed to the standard output.
Since boolean variable "isMarried" is not initialized, its value is by default false.

The correct answer is a.

Exercises
Declare a variable with the name **isForeigner** to know which workers are foreigners.
We assume that the most workers are foreigners.
Add a statement to the program to write the value of the variable **isForeigner** to the standard output.
Change the position of your previous statement in the code to see what happens.
Try to assign the new values 45, 298888888, 124.89, to the variables age, bank account, and wages. What is written to the standard output if the program is compiled and run?

Quiz 2: Primitive data types and variables

What happens when the following program is compiled and run?

```
class MyVariable
{
  byte b = 80;
  short s;
  float f1 = 3.50f;
  float f2;
  double d;

  public static void main(String[] args)
  {
    MyVariable mv = new MyVariable();

    System.out.print(mv.b + ", ");
    System.out.print(mv.s + ", ");
    System.out.print(mv.f1 + ", ");
    System.out.print(mv.f2 + ", ");
    System.out.print(mv.d);
  }
}
```

Select the correct answer:

a) This code writes "80, 0, 3.5, 0.0, 0.0" to the standard output.
b) This code writes "80, 0, 3.5, 0, 0" to the standard output.
c) This code writes "80, 0, 3.5, 0.0, 0" to the standard output.

Explanation
The default value of integers is "0", but the default value of floats and doubles are "0.0".

The correct answer is a.

Exercises
Assign the new values 122, 43.9f, 335.35 to the variables b, f2, d, and execute the program to see what happens.
Declare a character type variable called "myChar".
Assign the value "Q" to the variable "myChar".
Add a statement to the code to print the value of myChar to the standard output.
Change the position of your statement in the code to see what happens.

Quiz 3: Primitive data types default values

What happens when the following program is compiled and run?

```
class MyClass
{
  int i;
  double d;
  boolean b;

  public static void main(String[] args)
  {
    MyClass mc = new MyClass();

    System.out.print(mc.i + ", ");
    System.out.print(mc.d + ", ");
    System.out.print(mc.b);
  }
}
```

Select the correct answer:

a) This code writes "0, 0, false" to the standard output.
b) This code writes "0, 0.0, false" to the standard output.

Explanation
The correct answer is b, because the default value of double is "0.0".

The correct answer is b.

Exercises
Declare a variable called "myVar", and assign the value 1344.98 to it.
Declare a variable called "myVar2" directly under myVar, and assign the value "g" to it.
Declare a variable called "myVar3" directly under myVar2, and assign the value 766 to it.
Add three statements to the the program to write the values of myVar, myVar2 and myVar3 to the standard output.

Quiz 4: Assigning values to variables

What happens when the following program is compiled and run?

```
class MyClass
{
  int i1 = 7;
  int i2 = 12;

  public static void main(String[] args)
  {
    MyClass mc = new MyClass();

    mc.i1 = mc.i1 - 3;
    mc.i2 = mc.i2 + mc.i1;

    System.out.print(mc.i1 + ", ");
    System.out.print(mc.i2 + " ");
  }
}
```

Select the correct answer:

a) This code writes "7, 12" to the standard output.
b) This code writes "4, 19" to the standard output.
c) This code writes "4, 16" to the standard output.
d) This code writes "7, 19" to the standard output.
e) This program does not compile.

Explanation

i1 = i1 - 3 = 7- 3 = 4. the new value of i1 is 4, and that is why we use this value in the second equation, i2 = i2 + i1 = 12 + 4 = 16.

The correct answer is c.

Exercises

Add the statement "i1 = 9;" directly under the statement "public static void main(String[] args)".
Add the statement "i2 = 8;" directly under the previous statement.
What is written to the standard output if you compile and run the program?