ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/316997859

Easy Java: A Tutorial on Java Programming

Book · February 2015

CITATIONS
0

READS
31,959

1 author:

Abien Fred Agarap
De La Salle University
14 PUBLICATIONS **26** CITATIONS

SEE PROFILE

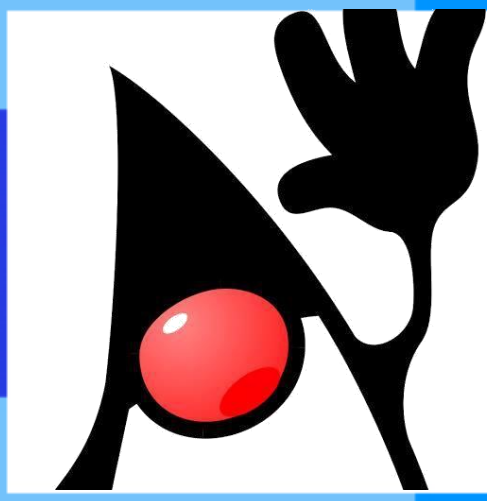
All content following this page was uploaded by Abien Fred Agarap on 18 May 2017.

The user has requested enhancement of the downloaded file.

EASY JAVA

Abien Fred Agarap
Intel ISEF 2013 Finalist

Preface



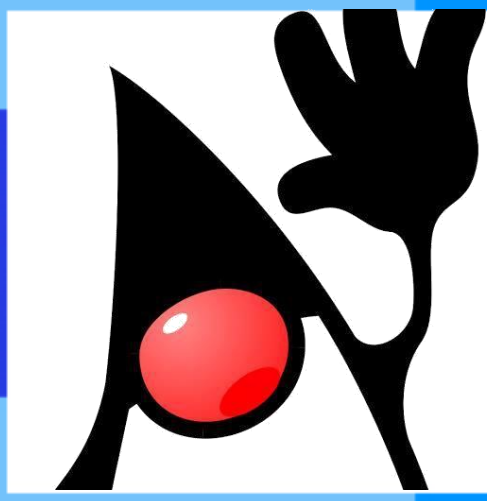
Easy Java

The world today is moving at an incredibly fast pace. This is why it is truly necessary that at a very young age, people are already equipped with the sophisticated knowledge to pursue their career path. So, *Easy Java* was written to motivate young students to pursue Computer Science. It will provide a *sneak peek* into the world of software development.

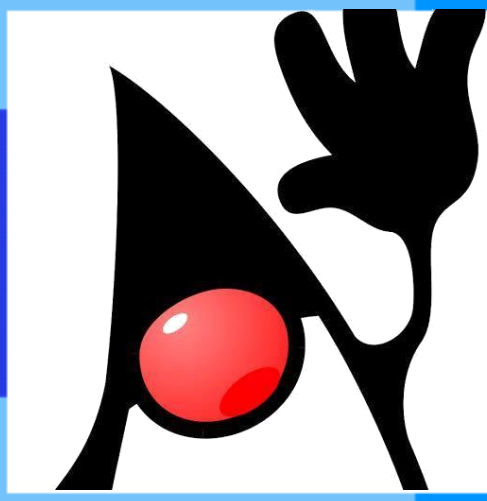
While this book is dominated by source codes, comments are provided for explanation of the syntaxes of the Java programming language.

The project files in this book can be downloaded at http://thereavisproject.webs.com.

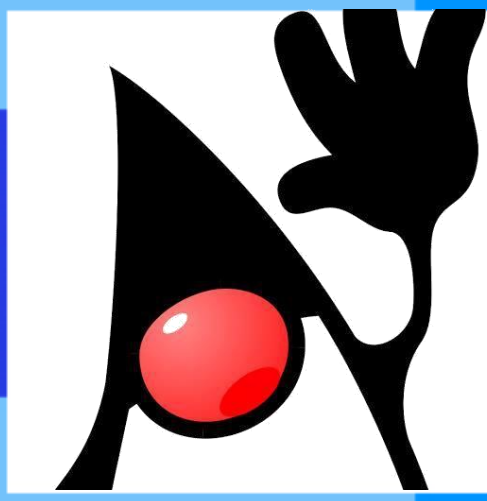
Contents



Contents



Introduction



Java Overview

What is Java?

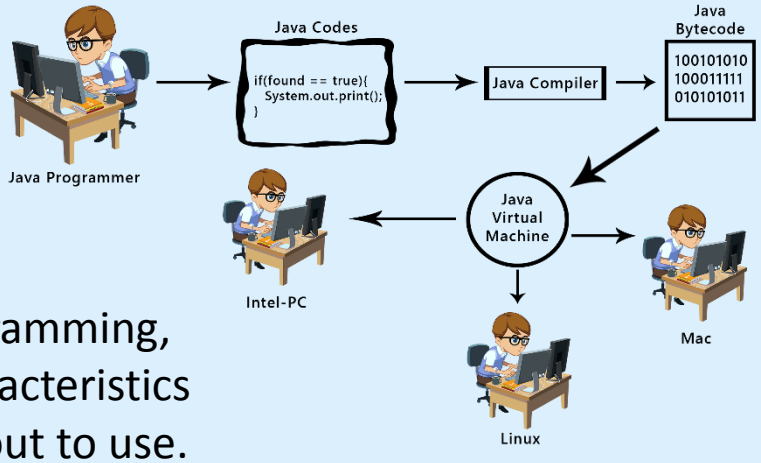
Java is an object-oriented programming language developed by Sun Microsystems as a core part of its Java platform which was released in 1995 (Lemay & Perkins, 1996; Java – Overview, n.d.). The development was led by James Gosling in 1991 (Horstmann, 2013; Java – Overview, n.d.; Java Fundamentals, n.d.). The language lets the developers to "write once, run anywhere" (WORA), and this gives Java the characteristic of being platform-independent (Java – Overview, n.d.; Java Fundamentals, n.d.). Its application to real world solutions ranges from personal computer up to mobile devices.

How does a Java program work?

Like other computer programs, Java source codes are translated into a machine language – binary language. This binary language is a set of zero's and one's that the computer only understands. However, there is a distinction to a Java-written program. Yes, it is translated into binary codes but it does not run directly under the computer system. Java programs run under the Java Virtual Machine installed in a computer system (*download and installation of Java Runtime Environment is needed*). For better understanding, refer to the illustration (Figure 1) on how a Java program compiles and runs.

Java Overview

Flowchart 1. How does a Java program work?



Characteristics of Java

Before diving straight into programming, it is important to know the characteristics of the language that we are about to use.

- **Object-oriented.** Everything in Java is an object.
- **Simple.** Easy to learn and use.
- **Platform-independent.** Java can execute in any computer system provided there is a Java run-time system present in it (Horstmann, 2013; Java – Overview, n.d.; Java Fundamentals, n.d.).
- **Architecture-neutral.** Java programs are not compiled for a specific processor and/or operating system (Java Fundamentals, n.d.). This is because Java programs run under the *Java Virtual Machine* or JVM (Java – Overview, n.d.).
- **Secure.** It enables to develop virus-free programs or software (Horstmann, 2013; Java – Overview, n.d.).
- **High performance.** Java uses Just-In-Time (JIT) compilers. A JIT compiler compiles code only if needed during an execution (Rouse, 2005).
- **Multi-threaded.** Java programs can perform several tasks simultaneously (Java – Overview, n.d.).
- **Case-sensitive.** In Java, “Name” is different from “name”.

Java Overview

First Sample Program

Download and install the Java Development Kit and NetBeans IDE. Then, you can create and run your first Java sample program in five simple steps. Necessary illustrations are provided for your easy understanding. The NetBeans IDE 8.0.2 will be used for developing the sample programs.

1. Launch NetBeans 8.0.2 and create a "New Project" by selecting "Java" in the Categories and "Java Application" in the Projects (Figure 1).

2. For the Project Name, enter "Chatbot"; for Project Location, choose any directory you would like to use. Then check the "Create Main Class". Click "Finish" afterwards (Figure 2).

3. Now that you have successfully created a new project, type these codes in the main method (Figure 3).

4. Run your first sample program by pressing *F6* in the keyboard or clicking the *Run* button in the NetBeans IDE.

5. As it would ask you your name, type your name in, and then press *Enter* in the keyboard. You would get this result (Figure 4).

Java Overview

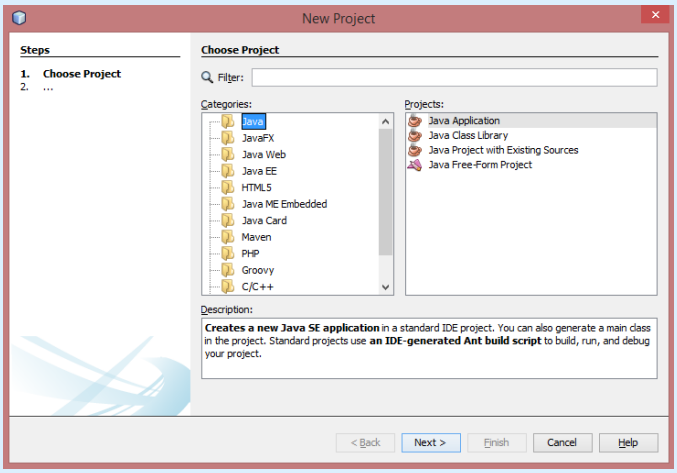


Figure 1. NetBeans 8.0.2: New Project Dialog Window

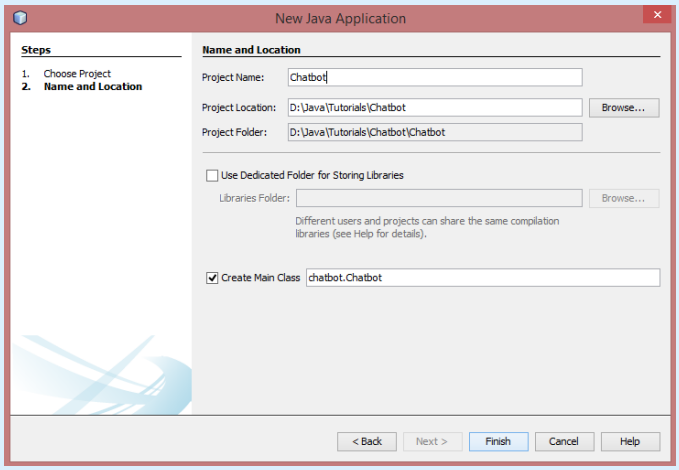


Figure 2. NetBeans 8.0.2: New Java Application Dialog Window

```java
package chatbot;

import java.util.Scanner;

public class Chatbot {

    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        String name;
        System.out.print("Who are you? ");
        name = in.nextLine();
        System.out.println("Hello, " + name + "!");
    }

}
```

Figure 3. Chatbot.java: Source code

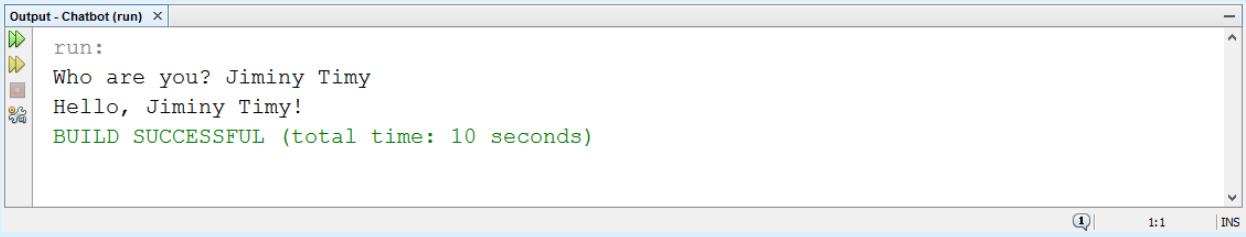


Figure 4. Chatbot.java: Output

Basic Syntax 1

Hello World (Console)

```
/*
    This is a comment, the Java compiler ignores this.
    Comments serve as documentation for your program.
    To write a comment, you may start either with
    double backslash (//) or how this comment
    is done backslash and asterisk.
*/
package helloworld;
/*
    Syntax: package <name>
    package groups related files.
*/
public class HelloWorld {
/*
    Class is the blueprint
    from which objects are created     */
```

Hello World (Console) – Cont.

```
public static void main(String[] args) {
        /*  This is the main method.
            It is the subprogram
            that the Java runs in your program.
        */
        System.out.println("Hello, World!");
        /*
            System.out.println is the method
            used to display text in the console.
            A method is a set of instructions
            that performs a particular task.
            Every statement in Java ends with a semicolon.
        */
    }
}
/*
    A set of braces {} depicts the beginning
    and the end of a class,
    a method, or a program. */
```

Hello World (GUI)

```
package helloworldgui;
import javax.swing.JOptionPane;
/*
   The import statement defines the reference to
   data and methods in other packages.
*/
public class HelloWorldGUI {

   public static void main(String[] args) {
      JOptionPane.showMessageDialog(null, "Hello, World!");
      /*
         Displays "Hello, World" in messagebox.
         The "showMessageDialog" is a method in
         the JOptionPane class. The IDE Intellisense
         can show you the parameters of this method.
         This sample is just for you to get
         starting on JOptionPane class if
         you wish to learn it.
      */
   }
}
```

Data Types

2

DataTypesIntroduction (Console)

```
package datatypesintroduction;
import java.util.Scanner;
/*  The Scanner class is used to
    receive input from the keyboard.    */
public class DataTypesIntroduction {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        /*  The "keyboard" object has been created from the
            Scanner class. Objects created from the Scanner class are
            used for accepting input from the keyboard. */
        System.out.print("Who are you? ");
        String name = keyboard.nextLine();
        /*  String is a sequence of characters
            To assign data for String, the text must be enclosed
            in double quote symbols.
                Syntax: String name = "Jones Kim";

            This case is an exception. The data is assigned directly
            from the keyboard input.    */
        System.out.println("Hello there, " + name + "!");
```

DataTypesIntroduction (Cont.)

```
 System.out.print("How old are you? ");
        byte age = keyboard.nextByte();
        /*  The byte is a number data type, it consumes 8 bits.
            The number you can store in byte is only up to 128. */
        System.out.println("Wow! You're just " + age + " years old?");
        System.out.print("Do you have books? How many books do
you have? ");
        short number_of_books = keyboard.nextShort();
        /*  The short is a number data type, it consumes 16 bits.
            The number you can store in short is only up to 256.    */
        System.out.println(number_of_books + "? Impresive.");
        System.out.print("Anyway, let's do some Math. Will you give
me two numbers? ");
        int num_one = keyboard.nextInt();
        /*  The int is the keyword for integer, it consumes 32 bits.
            This is the data type for whole numbers.
            The number you can store in int is from 2^-31 to 2^30
only. */
        double num_two = keyboard.nextDouble();
        /*  The double is a number data type that consumes 64 bits.
            This is the data type for numbers with decimal values.  */
        float product = (float) (num_one * num_two);
```

DataTypesIntroduction (Cont.)

```
 /*  The float is a data type for numbers with decimal values as
well.
          However, it only consumes 32 bits.  */
       /*  (float) (num_one * num_two); is an example of type
casting
          type casting is the process of converting one data type to
          another kind of data type.  */
       System.out.println("The product of " + num_one + " and " +
num_two + " is " + product);
       /*  Boolean is a data type that contains only two possible
values:
             a) true; b) false.
          It occupies only one bit, and its default value is false.
          Syntax:
             Boolean virus_found = true; */
       final double PI = 3.14159;
       /*  PI is a constant variable. The "final" keyword made
          it a constant variable. Writing constant variables
          in all uppercase letters is a convention.    */
       System.out.print("Give me a radius: ");
       double radius = keyboard.nextDouble();
       double circumference = 2 * PI * radius;
       System.out.printf("\nThe circumference of your circle is %.3f
cm\n", circumference);
 /*  The System.out.printf method displays
        Strings and other objects in a  specified format.   */
   }
}
```

Operators

3

ArithmeticOperators (Console)

```
package arithmeticoperators;
import java.util.Scanner;

public class ArithmeticOperators {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Enter first number: ");
        int num1 = keyboard.nextInt();
        System.out.print("Enter second number: ");
        int num2 = keyboard.nextInt();
        int ans = num1 + num2; // addition (+)
        System.out.println("\nSum of " + num1 + " and " + num2 + "
is " + ans);
        /*    The plus symbol (+) can be used for number addition or
String concatenation. */
```

Arithmetic Operators

ArithmeticOperators (Cont.)

```
 ans = num1 - num2; // subtraction (-)
        System.out.println("Difference of " + num1 + " and " + num2
+ " is " + ans);
        ans = num1 * num2; // multiplication (*)
        System.out.println("Product of " + num1 + " and " + num2 +
" is " + ans);
        ans = num1 / num2; // division (/)
        System.out.println("Quotient of " + num1 + " / " + num2 + "
is " + ans);
        ans = num1 % num2; // modulus (%)
        System.out.println("Remainder of " + num1 + " % " + num2 +
" is " + ans);
    }
}
```

Operators

4

AssignmentOperators (Console)

```
package assignmentoperators;
import java.util.Scanner;
public class AssignmentOperators {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Give me two numbers: ");
        int first_num = in.nextInt();
        int second_num = in.nextInt();
        /* The equal symbol (=) is used to assign the value
           of the right side to the variable in the left side.  */
        int total = first_num + second_num;
        System.out.println("Sum of two numbers: " + total + "\nGive
me an another number: ");
        int third_num = in.nextInt();
        total += third_num;
        //  The += symbol is a shortcut for a = a + b
        System.out.println("Result: " + total + "\nOkay, another one:
");
        int fourth_num = in.nextInt();
        total -= fourth_num;
```

Assignment Operators

AssignmentOperators (Cont.)

```
//  The -= symbol is a shortcut for a = a - b
        System.out.println(fourth_num + " has been subtracted from
the previous result: " + total
        + "\nGive me a number again: ");
        int fifth_num = in.nextInt();
        total *= fifth_num;
        //  The *= symbol is shortcut for a = a * b
        System.out.println("Result: " + total + "\nThis is the last time,
give me another number: ");
        int sixth_num = in.nextInt();
        total /= sixth_num;
        //  The /= symbol is a shortcut for a = a / b
        System.out.println("Result: " + total);
    }
```

Operators

5

RelationalOperators (Console)

```
package pkg5.relationaloperators;
import java.util.Scanner;
public class RelationalOperators {
    public static void main(String[] args) {
        /*  There is a relational operator in programming
            just like in Math. The following are the
            relational operators in Java programming:
                >=  Greater than or equal to
                <=  Less than or equal to
                ==  Equal to
                >   Greater than
                <   Less than
                !=  Not equal to
            There is no space between the two symbols in
            >=, <=, ==, and !=.
            The output of these operators is a Boolean       */
```

RelationalOperators (Cont.)

```
 System.out.print("Will you give two numbers? ");
        Scanner in = new Scanner (System.in);
        int first_num = in.nextInt();
        int second_num = in.nextInt();
        System.out.println(first_num + " is greater than " +
second_num + ": " +(first_num > second_num));
        System.out.println(first_num + " is less than " + second_num
+ ": " +(first_num < second_num));
        System.out.println(first_num + " is equal to " + second_num
+ ": " +(first_num == second_num));
        System.out.println(first_num + " is not equal to " +
second_num + ": " +(first_num != second_num));
        System.out.println(first_num + " is greater than or equal to "
+ second_num + ": " +(first_num >= second_num));
        System.out.println(first_num + " is less than or equal to " +
second_num + ": " +(first_num <= second_num));
    }
```

Operators

6

LogicalOperators (Console)

```
package pkg6.logicaloperators;
import java.util.Scanner;
public class LogicalOperators {
    public static void main(String[] args) {
        /*  Logical operators are used for testing multiple conditions to arrive at one
decision.
            The logical operators in Java are the following:
                &&  (And)   --  All expressions must be true to have a TRUE result.
                ||  (Or)    --  At least one expression must be true to have a TRUE result.
                !   (Not)   --  Expression must not be equal to a given value to have a TRUE
result. */
        Scanner in = new Scanner(System.in);
        System.out.print("Enter two numbers: ");
        int first_num = in.nextInt();
        int second_num = in.nextInt();
        System.out.println("First number is equal to 10 AND second number is equal to
20? " + ((first_num == 10) && (second_num == 20)));
        System.out.println("First number is equal to 10 OR second number is equal to
20? " + ((first_num == 10) || (second_num == 20)));
        System.out.println("First number is NOT equal to 10? " + (!(first_num == 10)));
        System.out.println("Second number is NOT equal to 20? " + (!(second_num ==
20)));
    }
}
```

Math Methods 7

MathMethods (Console)

```
package pkg7.mathmethods;
import java.util.Scanner;
public class MathMethods {
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        System.out.print("Let's apply some Math methods to a number: ");
        int number = in.nextInt();
        System.out.println("Square root of " + number + " is equal to " +
Math.sqrt(number));
        System.out.println("Absolute value of " + number + " is equal to " +
Math.abs(number));
        System.out.println("Radians value of " + number + " is equal to " +
Math.toRadians(number));
        System.out.println("Cosine " + number + " is equal to " +
Math.cos(Math.toRadians(number)));
        /*  For trigonometric functions, it assumes the given value
            is in its radian form. Math.sin(30) will not yield 0.5.
            It must be Math.sin(Math.toRadians(30)).    */
        System.out.print("\nGive me an another number: ");
        int base = in.nextInt();
        System.out.print("Give me a number to serve as an exponent: ");
        int exponent = in.nextInt();
        System.out.println(base + " raise to " + exponent + " is equal to " +
Math.pow(base, exponent));
    }
}
```

Decision

8

DecisionStatements (Console)

```
package pkg8.decisionstatements;
import java.util.Scanner;
/*  This is a program to compute the benefits of an employee based on his/her
number of years in the company. This also computes how much will the employee
receive for his/her bonus.
    Inputs are:
        1. Number of years
        2. Number of hours worked in a month    */
public class DecisionStatements {
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        System.out.print("Enter years of service in the company: ");
        int years = in.nextInt();
        /*  The switch statement is used to check for multiple condition for a single
value. Each condition is called a "case". Cases must have a "break" to serve as a
separator. Without break, once a condition is true in a particular case, its body and
also the body of other cases will run. The default statement which is used to contain
any statement you want to execute in the event that not a single one of the
conditions in the switch statement is met. The default statement is optional, but it is
recommended for user-friendliness. Default case does require to have a break.
                Syntax:
                    switch (expression) {
                        case <condition1>:  // body
                                        break;
                        case <condition2>: // body
                                        break;      */
```

DecisionStatements (Cont.)

```
switch (years) {
        case 0: System.out.println("No benefits yet!"); break;
        case 1:
        case 2:
        case 3:
        case 4:
        case 5: System.out.println("Free car!"); break;
        case 6:
        case 7:
        case 8:
        case 9:
        case 10:
        case 11:
        case 12:
        case 13:
        case 14:
        case 15: System.out.println("Free house and lot!"); break;
        default: System.out.println("Error!");
    }
    System.out.print("Enter number of hours worked in a month: ");
    int hours = in.nextInt();
    /*  The if statement is used to check for a condition.
        If the condition is true, its body will be executed.
            Syntax: if (condition) { // body }
        The if...else if statement is used to check for multiple conditions.
        This construct is more efficient than having multiple if statements
        to check for a condition for a single variable or value.
            Syntax: if (condition) { // body } else if (condition) { // body }
            Note:   An else statement is option in if...else if statement.
                    Multiple statements must be enclosed using
                    a set of braces.    */
```

DecisionStatements (Cont.)

```
if (hours == 176) {
        /*  Assuming there are 22 working days in a month,
            and 8 working hours in a day, 176 is just exact.    */
        System.out.println("You will receive an extra $250 as a bonus.");
    }
    else if (hours > 176 && hours <= 242) {
        /*  Assuming that the allowed over-time
            is up to three hours only, 242 is
            the maximum number of hours if the
            employee is working over-time (for three hours)
            in 22 working days of the month.    */
        System.out.println("You will receive an extra $1500 as a bonus.");
    }
    else if (hours < 176 && hours >= 0) {
        //  Employee is under-time
        System.out.println("You are fired.");
    }
    else {
        //  This else is for an error message
        System.out.println("Error!");
    }
  }
}
```

Loops

9

LoopStatements (Console)

```
package pkg9.loopstatements;
import java.util.Scanner;
/*  Looping statements and Arrays will be introduced in this source
code.
    Decision statements will be incorporated in this program.

    This program will accept scores as an input. This program will
    determine if the score is a failing score or not.   */
public class LoopStatements {
    public static void main(String[] args) {
        System.out.print("Enter the perfect score: ");
        Scanner keyboard = new Scanner(System.in);
        int perfect_score = keyboard.nextInt();
        /*  Array is a set or series of objects that have the same size and
type.

            The counting of array elements starts
            from 0 up to 1 less than the
            size of the array.

                Syntax:
                    int [] <array_name> = new int [<array_size>];   */
```

LoopStatements (Cont.)

```
 /*  The while loop is used to repeat a statement or series of statements as
long as a given condition is true.

          Syntax:
             while (condition) {
                // loop body
             }   */
      int [] scores = new int [0];
      while (true) {
         System.out.print("How many scores are you going to enter? ");
         byte index = keyboard.nextByte();
         if (index >= 1){ scores = new int [index]; break; }
      }

      /*
         The do...while loop is similar to
         the while loop. It is used when
         you want to execute the loop
         body at least once.

             Syntax:
                do{
                   // loop body
                }while(condition);
      */
```

LoopStatements (Cont.)

```
int h = 0;
      do{
         System.out.print("Enter score: ");
         scores[h] = keyboard.nextInt();
         h++;
      }
      while(h < scores.length);
      /*
         The length method gets the value
         of the array's index.
      */

      /*
         The for loop is used when the
         developer knows the specific number of
         times he/she wants to repeat the loop body.

            Syntax:
               for (<initialization_of_loop_counter>; <condition>;
<increment/decrement>){
                  // loop body
               }

         Increment operator (++) is short for x = x + 1
         Decrement operator (--) is short for x = x - 1
      */
```

LoopStatements (Cont.)

```
for (int i = 0; i < scores.length; i++) {
        /*
            If the score is from 0 to the less than
            70% of the perfect score          */
        if (scores[i] < (perfect_score * 0.70) && scores[i] >= 0){
            System.out.println(scores[i] + ": FAILED.");
        }
        /*
            If the score is from the 70% of the
            perfect score and the score is
            less than the perfect score.
        */
        else if (scores[i] >= (perfect_score * 0.70) && scores[i] <
perfect_score){
            System.out.println(scores[i] + ": PASSED.");
        }
        //  If the score is a perfect score
else if (scores[i] == perfect_score){
            System.out.println(scores[i] + ": PERFECT.");
        }
/*          This else is written to display
            a message if the score entered
            is greater than the perfect score
            or if the score is less than 0.     */
        else{
            System.out.println("ERROR!");
        }
    }
  }
```

Arrays

10

Arrays (Console)

```
package pkg10.arrays;
import java.util.Scanner;
import static java.lang.System.out;
/*  More about arrays will be discussed in this program.

    This program will ask for the student number as an input.
    After entering the student number, it will display the
    student with that number.   */
public class Arrays{
    public static void main (String[]args){
        Scanner in = new Scanner (System.in);
        String [] students = {"Mark Horstmay", "Rose Daud", "Steven Adam", "Lucas
Alan", "May Wan"};
        System.out.print("Enter student number: ");
        String student_no = in.nextLine();
        switch (student_no){
            /*  You may acces an array element by using the array element number.
                <array_name> [<index_number>];  */
            case "201510101565": out.println("Name of student: " + students[0]); break;
            case "201510101566": out.println("Name of student: " + students[1]); break;
            case "201510101567": out.println("Name of student: " + students[2]); break;
            case "201510101568": out.println("Name of student: " + students[3]); break;
            case "201510101569": out.println("Name of student: " + students[4]); break;
            default: out.println("NOT FOUND IN DIRECTORY!");
        }
    }
}
```

11 Methods

Methods (GUI)

```
package pkg11.methods;
import javax.swing.JOptionPane;
/*
    In your HelloWorld.java program,
    the term method has been introduced.

    Method is a set of instructions
    defined to accomplish a specific task.

    There are two kinds of Methods:
        1. Procedures - they do not return a value
        2. Functions - they return a value

    To quickly identify a procedure,
    it is always declared with
    the "void" keyword.
        e.g. void displayMessage()

    On the other hand, functions
    are declared with other
    data types.
        e.g. int addNumbers(int x, int y)
*/
```

Methods (Cont.)

```
public class Methods {

    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "This program will add two
numbers.");
        int first_number =
Integer.parseInt(JOptionPane.showInputDialog(null, "Enter first number:"));
        int second_number =
Integer.parseInt(JOptionPane.showInputDialog(null, "Enter second
number:"));
        /*
            Integer.parseInt was used to convert the value received
            from the JOptionPane.showInputDialog.

            The JOptionPane.showInputDialog gives a String value
            as an output, that is why converting the data
            using Integer.parseInt was necessary.
        */
        int total = addNumbers(first_number, second_number);
        JOptionPane.showMessageDialog(null, "The sum of " + first_number
+ " and " + second_number + " is " + total + ".");
        JOptionPane.showMessageDialog(null, "This program will now
multiply two numbers.");
        /*
            Now that we are done using the first_number
            and second_number variables for receiving
            numbers to add, we can now re-use them.   */
```

Methods (Cont.)

```
 first_number = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter first number:"));
        second_number = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter second number:"));
        multiplyNumbers(first_number, second_number);
    }

    /*
        Parts of a method:

        <access_type> <return_type> <method_name> (<data_type> <parameter_name>){
            // method body
        }

        For multiple parameters, parameters must be
        separated by using comma.
    */
    public static int addNumbers(int first_no, int second_no){
        return first_no + second_no;
    }
```

Methods (Cont.)

```
 /*
        The addNumbers is designed to receive two parameters:
            1. first_no
            2. second_no

        It does not matter if the parameter name (formal parameter)
        in the method declaration matches the parameter name
(actual parameter)
        in the method call. However, they must be arranged
accordingly.
    */

    /*
        multiplyNumbers is a procedure.
        The statement to display the multiplication
        result is written within its body.
    */
    public static void multiplyNumbers(int first_no, int second_no){
        int product = first_no * second_no;
        JOptionPane.showMessageDialog(null, "The product of " +
first_no + " and " + second_no + " is " + product + ".");
    }
}
```

12 Classes & Objects

ClassesAndObjects (Console)

```
package pkg12.classesandobjects;
import java.util.Scanner;
class Animal{
/*  As explained in the HelloWorld application, class is the blueprint of your
program from where objects are created.

    For instance, we have the Animal class.
    This class could have objects such as move, eat, and color among others.

    An object could be further divided into
    two: 1) Methods; 2) Properties

    Actually, you already know those two.
    Methods are a set of instructions that accomplish a task;
    Properties are the data (String, number).   */
    String name;
    void setName(String name){
        this.name = name;
        /*  The term "this" is used to refer to the object's data
            if the formal parameter name and the object name is identical.          */
    }
    void displayName(){
        System.out.println("Animal name: " + name);
    }
}
```

ClassesAndObjects (Cont.)

```
public class ClassesAndObjects {
    public static void main(String[] args) {
        System.out.println("Creating octopus...");
        Animal octopus = new Animal();
        /*  The "octopus" has been created from the Animal class.
            This is the syntax for creating objects from classes:
                Syntax:
                    <class_name> <object_name> = new <constructor_name>();
            Don't bug yourself with the contructor name yet.
            For now, just type in the class name for the constructor name.
        */
        System.out.print("Name your octopus: ");
        Scanner in = new Scanner(System.in);
        String name = in.nextLine();
        //  The "name" object contains the string that the user will enter.
        octopus.setName(name);
        /*  In the Animal class, the method setName is used to assign the
actual parameter
            to its "name" object.

            To access the methods or data of an object, use the object name,
            followed by a dot (.), then the method or variable that you want
            to call or use in the program.
        */
        octopus.displayName();
        /*  After setting a name for the animal, the displayName() method
            shall now display the name for the animal you have created.
        */
    }
```

Examples 13

Dice (Console)

```
package pkg13.dice;
import java.util.Scanner;
import static java.lang.System.out;
public class Dice{
        public static void main(String[] args) {
                Scanner in = new Scanner (System.in);
                out.println("DICE SIMULATION");
                byte guess;
                while(true){
                out.print("Enter your bet (1-6 only): ");
                guess = in.nextByte();
                if (guess > 0 && guess < 7) break;
                }
                out.println("\nRolling dice...");
```

Dice (Cont.)

```
            byte dice = (byte) ((Math.random() * 6) + 1);
/*  The Math.random() method is used to generate a
random number.

Six was used as a multiplier for generating the computer
dice to tell the computer that the random number must be
limited up to number 6 only. An additional one was written
for instances when the generated random number is zero.
This way, it will always produce a number greater than 0,
but less than or equal to 6.
*/
            System.out.println("\nDice: " + dice);
            if (guess == dice) out.println("You win!");
            else out.println("You lose!");
            }
}
```

GetSalePrice (Console)

```
package pkg13.getsaleprice;
import java.util.Scanner;
public class GetSalePrice{
    public static void main(String[] args) {
        double regular_price = get_regular_price();
        // regular_price will contain the value returned by the get_regular_price()
function.
        double sale_price = regular_price - getDiscountedPrice(regular_price);
        /*  sale_price will contain the difference between the
            regular_price and the value returned by the getDiscountedPrice
function.
        */
        System.out.printf("Discounted price: Php %.2f", sale_price);
    }
    /*  Methods or variables that are static are those that can only have a single
                instance unlike the non-static methods/variables. */
    static double get_regular_price(){
        Scanner in = new Scanner (System.in);
        System.out.print("Enter item's regular price: Php ");
        return in.nextDouble();
        /*  Instead of declaring another variable for the return value,
            This method will directly return the input of the user
            to the calling statement (double sale_price = get_regular_price()).
            The only problem with this strategy is if the user enters a negative or a
            zero value.      */
    }
    static double getDiscountedPrice(double price){
        final double DISCOUNT_PERCENTAGE = 0.20;
        return price * DISCOUNT_PERCENTAGE;
    }
}
```

CoinToss (Console)

```
package pkg13.cointoss;
import java.util.Scanner;
public class CoinToss {
    public static void main(String[] args) {
        System.out.println("COIN TOSS SIMULATION");
        System.out.println("[1] Heads\t[2] Tails");
        if ((byte)(Math.random() * 2 + 1) == get_bet())
                System.out.println("You win!");
        else System.out.println("You lose!");
}
    static byte get_bet(){
        Scanner in = new Scanner (System.in);
        System.out.print("Enter your bet: ");
        return in.nextByte();
    }
}
```

CommissionRate (Console)

```
package pkg13.commissionrate;
import java.util.Scanner;
//  This program calculates the salesperson's pay
public class CommissionRate{
    public static void main(String[] args) {
        // Get the value of sales
        double sales = get_sales();

        // Get the value of advanced pay
        double advanced_pay = get_advanced_pay
```

CommissionRate (Cont.)

```
        // Determine the commission rate based on monthly sales
        double comm_rate = determine_comm_rate(sales);

        // Calculate the pay
        double pay = sales * comm_rate - advanced_pay;
        System.out.printf("The pay is Php %.2f", pay);

        // Check if the pay is negative
        if (pay < 0) System.out.println("The salesperson must reimburse the
company.");
    }
    /*  Scanner is declared static. Thus, eliminating the need
        to create a Scanner object in each and every method
        that needs it. The Scanner object can be accessed
        by all methods in this program. */
    static Scanner in = new Scanner (System.in);
    static double get_sales(){
        System.out.print("Enter monthly sales: ");
        return in.nextDouble();
    }
    static double get_advanced_pay(){
        System.out.print("Enter advanced pay: ");
        return in.nextDouble();
    }
    static double determine_comm_rate(double sales){
        if (sales < 10000) return 0.10;
        else if (sales >= 10000 || sales <= 14999) return 0.12;
        else if (sales >= 15000 || sales <= 17999) return 0.14;
        else if (sales >= 18000 || sales <= 21999) return 0.16;
        else if (sales >= 22000) return 0.20;
        else return 0;
    }
}
```

CellphoneObject (Console)

```
package pkg13.cellphone;
import java.util.Scanner;
class Cellphone{
    String model, price;
    /*  When an object is created, Java calls the constructor first.
        Thus, eliminating the need for manually invoking the
constructor.
        Any code in the constructor will then be executed.
        If there is no constructor written, Java creates one
        for the program. But by default, that constructor is blank.

        Constructor name must be identical to its class name.
        Constructors are neither function or procedure.
        Therefore, it does not need method type keywords
        (void, double, int, etc.)   */
    Cellphone(String model, String price){
        /*  This constructor only assigns the actual parameters
            from the invoking statement to the data of the
            Cellphone class: 1. model; 2. price
        */
        this.model = model;
        this.price = price;
    }
```

CellphoneObject (Cont.)

```
 String get_model(){
        return this.model;
    }
    String get_price(){
        return this.price;
    }
}
public class CellphoneObject {
    public static void main(String[] args){
        Scanner in = new Scanner (System.in);
        System.out.print("Enter phone model: ");
        String model = in.nextLine();
        System.out.print("Enter phone price: Php ");
        String price = in.nextLine();
        /*  price is declared String because there
            will be no computation in this program.  */
        System.out.println();
        Cellphone phone = new Cellphone(model, price);
        System.out.println("Phone model: " + phone.get_model());
        System.out.println("Phone price: Php " + phone.get_price());
    }
}
```

MiniCashierProgram (GUI)

```
package pkg13.minicashierprogram;
import javax.swing.JOptionPane;
class Cashier{
    int display_menu(){
        return Integer.parseInt(JOptionPane.showInputDialog(null, "Choose
item to order:"
                + "\n[1] Diamond necklace ($10,000)"
                + "\n[2] Silver ring ($20)"
                + "\n[3] iPhone 6 Plus ($800)"));
    }
    void pay_item(int choice){
        double amount_due = 0, change;
        switch(choice){
            case 1: amount_due = 10000; break;
            case 2: amount_due = 20; break;
            case 3: amount_due = 800; break;
        }
        double cash =
Double.parseDouble(JOptionPane.showInputDialog(null, "Enter your
payment:"));
        if (cash > 0){
            change = cash - amount_due;
            JOptionPane.showMessageDialog(null, "Amount Due: $" +
amount_due + "\nCash: $" + cash + "\nChange: $" + change);
        }
        else JOptionPane.showMessageDialog(null, "Insufficient amount!");
    }
}
```

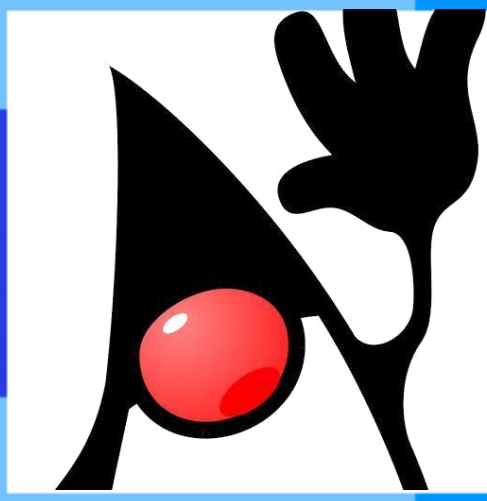
MiniCashierProgram (Cont.)

```
public class MiniCashierProgram {
    public static void main(String[] args) {
        Cashier a = new Cashier();
        int choice = 0;

        /*  Loop the menu as long as the user is not entering
            a correct choice (1-3 only).    */
        do{
            choice = a.display_menu();
        }while(choice <= 0 || choice > 3);

// Pay for the chosen item
        a.pay_item(choice);
    }
}
```

Exercises



Easy

1. Write a Java program that will display a similar output to the following (use looping statements):

FEBRUARY

SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

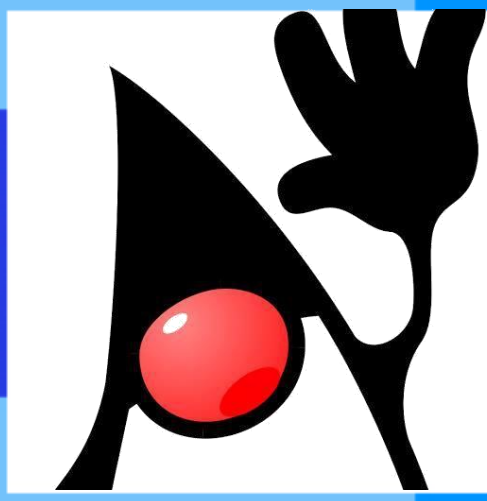
2. Get string as an input, display a version of the input without the first three characters. [remove("Star"); → "r"]

3. Look for letter "m" in a string. If the said letter is present in the string, return true. Otherwise, return false. [checkForM("Me"); → True]

4. Input an array of scores. If the scores are ordered in decreasing order, return true. Otherwise, return false. [decreasingNum(5, 4, 3); → True] The array should have 3 or more elements.

5. Make a program that will determine if the input number is odd or even. [oddOrEven(5) → "odd"]

Exercises



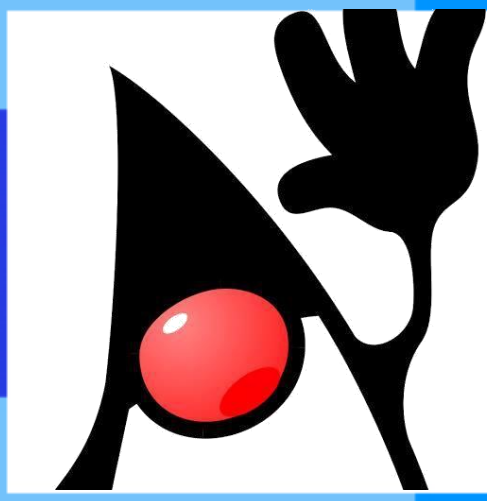
Average

1. Make a program that will return the product of the digits (0-9) that appear in a string. Ignore all characters other than the numbers. Return 0 if there are no digits in the string. [productNumbers(“hello123”) → 6]

2. Write a program that will get the 6 trigonometric functions of a given angle.

3. Input numbers in an array. Display the inputted array elements in reverse order.

Exercises



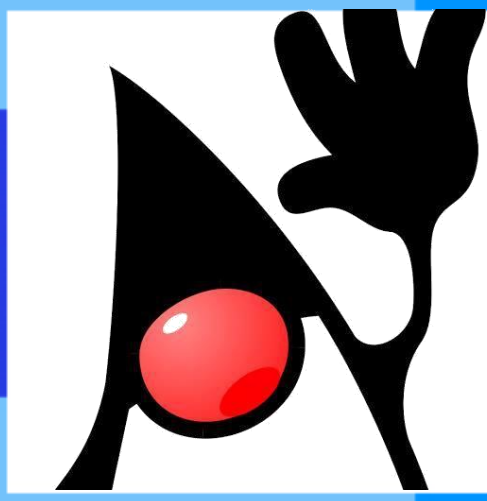
Challenge

1. Write a program that will write your name in a text file.

2. Design and create a GUI program that will play a pre-selected music.

3. Make a program that will delete all the *.db files in a certain directory.

References



Easy Java

Gaddis, T. (2012). *Starting out with Python* (2nd ed.). United States of America: Pearson Education, Inc.

Horstmann, C. (2013). *Java for everyone, late objects* (2nd ed.). Jefferson City: John Wiley & Sons, Inc.

Lemay, L., & Perkins, C. L. (1996). *Teach yourself Java in 21 days.* Indianapolis: Sams.net Publishing.

Java Fundamentals. (n.d.). Java fundamentals. *Oracle.* Retrieved from http://www.oracle.com/events/global/en/java-outreach/resources/java-a-beginners-guide-1720064.pdf.

Java – Overview. (n.d.). Java – Overview. *Tutorialspoint.* Retrieved from http://www.tutorialspoint.com/java/java_overview.htm.

Rouse, M. (2005, April). Just-in-time compiler (JIT). *WhatIs.com.* Retrieved from http://whatis.techtarget.com/definition/just-in-time-compiler-JIT.

Java logo: http://www.logospike.com/wp-content/uploads/2014/11/Java_logo-3.gif