Motivations

In the preceding chapter, you learned how to create, compile, and run a Java program. Starting from this chapter, you will learn how to solve practical problems programmatically. Through these problems, you will learn Java primitive data types and related subjects, such as variables, constants, data types, operators, expressions, and input and output.
Programming Example: Area of A Circle

```java
public class ComputeArea {
    public static void main(String[] args) {
    }
}
```
Compute Area of A Circle

```java
public class ComputeArea {
    public static void main(String[] args) {
    }
}
```

Main method is always named “main” so, it indicates the start point of the program
Identifiers

public class ComputeArea {

}

ComputeArea is an Identifier.

Identifiers are names that identify the elements such as classes, methods, and variables.
public class ComputeArea {
    public static void main(String[] args) {

        // Define inputs and outputs
        // input : radius, output : area

        // assign values to inputs and (outputs if necessary)

        // calculate area based on radius
        // show it to user
    }
}
public class ComputeArea {

    public static void main(String[] args) {
        //Define inputs and outputs
        //input : radius, output : area

        //assign values to inputs and (outputs if necessary)

        //calculate area based on radius
        //show it to user
    }
}
public class ComputeArea {

    public static void main(String[] args) {

        //Define inputs and outputs
        //input : radius, output : area

        //assign values to inputs and (outputs if necessary)

        //calculate area based on radius

        //show it to user
    }
}
public class ComputeArea {

    public static void main(String[] args) {

        //Define inputs and outputs
        //input : radius, output : area

        //assign values to inputs and (outputs if necessary)

        //calculate area based on radius

        //show it to user

    }

}
//Define inputs and outputs
//input : radius, output : area

What is radius? (What values it can hold?)
What is area? (What values it can hold?)
How many memory cells are required for radius?
How many memory cells are required for area?
Variables

radius and area are variables!

A variable is an identifier that it's value might change during the program.
Defining Variables

To define a variable we need to specify their context (type of values it can hold and the required memory size) using a datatype.

```java
datatype variableName;
```

This statement will allocate memory in RAM and will name it `variableName`. 
public class ComputeArea {

    public static void main(String[] args) {
        //Define inputs and outputs
        //input : radius, output : area
        double radius;
        double area;

        //assign values to inputs and (outputs if necessary)

        //calculate area based on radius

        //show it to user
    }
}
public class ComputeArea {

    public static void main(String[] args) {
        //Define inputs and outputs
        //input : radius, output : area
        double radius;
        double area;

        //assign values to inputs and (outputs if necessary)
        radius = 20;

        //calculate area based on radius
        //show it to user
    }
}
Initialization

The first time we assign a value to a variable, we are initializing it!

In Java, no variable can be used without initialization!
public class ComputeArea {
    public static void main(String[] args) {
        // Define inputs and outputs
        // input: radius, output: area
        double radius;
        double area;

        // assign values to inputs and outputs if necessary
        radius = 20;

        // calculate area based on radius
        area = radius * radius * 3.14159;

        // show it to user
    }
}
Assignment

VariableName = Expression;

It doesn't mean equality, it means calculate the expression in the right and assign it to the variableName in the left!
public class ComputeArea {
    public static void main(String[] args) {
        //Define inputs and outputs
        //input : radius, output : area
        double radius;
        double area;

        //assign values to inputs and (outputs if necessary)
        radius = 20;

        //calculate area based on radius
        area = radius * radius * 3.14159;

        //show it to user
        System.out.println("The area for the circle of radius " + radius + " is " + area);
    }
}
Printing in the command line

System.out.println(What_We_Want_To_Print);

We can print a word, a sentence, a variable, or numbers concatenated by plus sign (+).

System.out.println("The area for the circle of radius " + radius + " is " + area);
public class ComputeArea {

    public static void main(String[] args) {
        //Define inputs and outputs
        //input : radius, output : area
        double radius;
        double area;

        //assign values to inputs and (outputs if necessary)
        radius = 20;

        //calculate area based on radius
        area = radius * radius * 3.14159;

        //show it to user
        System.out.println("The area for the circle of radius " + radius + " is " + area);
    }
}
public class ComputeArea {
    public static void main(String[] args) {
        // Define inputs and outputs
        // input: radius, output: area
        double radius;
        double area;

        // assign values to inputs and (outputs if necessary)
        radius = 20;

        // calculate area based on radius
        area = radius * radius * 3.14159;

        // show it to user
        System.out.println("The area for the circle of radius " + radius + " is " + area);
    }
}
public class ComputeArea {
    public static void main(String[] args) {
        // Define inputs and outputs
        // input: radius, output: area
        double radius;
        double area;

        // assign values to inputs and (outputs if necessary)
        radius = 20;

        // calculate area based on radius
        area = radius * radius * 3.14159;

        // show it to user
        System.out.println("The area for the circle of radius " + radius + " is " + area);
    }
}
public class ComputeArea {
    public static void main(String[] args) {
        // Define inputs and outputs
        // input: radius, output: area
        double radius;
        double area;

        // assign values to inputs and (outputs if necessary)
        radius = 20;

        // calculate area based on radius
        area = radius * radius * 3.14159;

        // show it to user
        System.out.println("The area for the circle of radius "+radius+" is "+area);
    }
}
public class ComputeArea {
    public static void main(String[] args) {
        // Define inputs and outputs
        // input: radius, output: area
        double radius;
        double area;

        // assign values to inputs and (outputs if necessary)
        radius = 20;

        // calculate area based on radius
        area = radius * radius * 3.14159;

        // show it to user
        System.out.println("The area for the circle of radius " + radius + " is " + area);
    }
}
Named Constants

**Named Constant** is an *identifier* that represents a permanent value.

```java
final datatype CONSTANTNAME = VALUE;
```

```java
final double PI = 3.14159;
```
public class ComputeArea {
    public static void main(String[] args) {
        final double PI = 3.14159;

        //Define inputs and outputs
        double radius;
        double area;

        //assign values to inputs and (outputs if necessary)
        radius = 20;

        //calculate area based on radius
        area = radius * radius * PI;

        //show it to user
        System.out.println("The area for the circle of radius "+radius+" is "+area);
    }
}
Naming Identifiers

An identifier is a sequence of characters that consist of letters, digits, underscores (_), and dollar signs ($).

An identifier must start with a letter, an underscore (_), or a dollar sign ($). It cannot start with a digit.
- An identifier cannot be a reserved word. (See Appendix A, “Java Keywords,” for a list of reserved words).

An identifier cannot be true, false, or null.

An identifier can be of any length.
Declaring and Initializing in One Step

```java
int x = 1;
double d = 1.4;
```
Naming Conventions

Choose meaningful and descriptive names.

Variables and method names:

- Use lowercase. If the name consists of several words, concatenate all in one, use lowercase for the first word, and capitalize the first letter of each subsequent word in the name. For example, the variables `radius` and `area`, and the method `computeArea`.
Naming Conventions, cont.

Class names:
- Capitalize the first letter of each word in the name. For example, the class name `ComputeArea`.

Constants:
- Capitalize all letters in constants, and use underscores to connect words. For example, the constant `PI` and `MAX_VALUE`
Numerical Data Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>Storage Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>$-2^7$ to $2^7 - 1$ (-128 to 127)</td>
<td>8-bit signed</td>
</tr>
<tr>
<td>short</td>
<td>$-2^{15}$ to $2^{15} - 1$ (-32768 to 32767)</td>
<td>16-bit signed</td>
</tr>
<tr>
<td>int</td>
<td>$-2^{31}$ to $2^{31} - 1$ (-2147483648 to 2147483647)</td>
<td>32-bit signed</td>
</tr>
<tr>
<td>long</td>
<td>$-2^{63}$ to $2^{63} - 1$ (i.e., -9223372036854775808 to 9223372036854775807)</td>
<td>64-bit signed</td>
</tr>
</tbody>
</table>
| float | Negative range: $-3.4028235E+38$ to $-1.4E-45$  
Positive range: $1.4E-45$ to $3.4028235E+38$ | 32-bit IEEE 754 |
| double | Negative range: $-1.7976931348623157E+308$ to $-4.9E-324$  
Positive range: $4.9E-324$ to $1.7976931348623157E+308$ | 64-bit IEEE 754 |
Numerical Data Types: byte

byte
- An integer number which can be stored in one byte
- The number can be negative or positive (signed)
- One byte can store 256 numbers

\[-128, -127, \ldots, 0, 1, \ldots, 127\]
\[-2^7 \text{ to } 2^7-1\]
Numerical Data Types: short

**short**
- An integer number which can be stored in two bytes (16 bits)
- The number can be negative or positive (signed)

[-2^{15} to 2^{15}-1]

short a = 100;
Numerical Data Types: int

- An integer number which can be stored in 4 bytes (32 bits)
- The number can be negative or positive (signed)
Numerical Data Types: long

long

- An integer number which can be stored in 8 bytes (64 bits)
- The number can be negative or positive (signed)
Numerical Data Types

byte a = 10;
short b = 300;
int c = 46000;
long d = 567837927347573;

Select the one that is most appropriate for your variable.

For simplicity will use int most of the time.
Numerical Data Types: float

float

- A floating-point number which can be stored in 4 bytes (32 bits)

- The number can be negative or positive (signed)
Numerical Data Types: double

double
- A floating-point number which can be stored in 8 bytes (64 bits)
- The number can be negative or positive (signed)
Numerical Data Types

float a = 3.59374736;
Double b = 5.3e-200;
- float is known as single precision
- double is known as double precision

- double is more accurate than float

- We will use double most of the time
Numerical Data Types: Overflow

```
int a = 10;
int a = -10;
int a = 2147483648;
```

Java Compile Error: integer number too large: 2147483648;
Numerical Data Types: Overflow

```java
int a = 2147483647 + 1;
```

No Compile Error! But the value will be stored incorrectly!

Value actually will be: -2147483648

Be careful!
# Numeric Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>34 + 1</td>
<td>35</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>34.0 - 0.1</td>
<td>33.9</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>300 * 30</td>
<td>9000</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>1.0 / 2.0</td>
<td>0.5</td>
</tr>
<tr>
<td>%</td>
<td>Remainder</td>
<td>20 % 3</td>
<td>2</td>
</tr>
</tbody>
</table>
Remainder Operator

Remainder is very useful in programming.

- An even number % 2 is always 0 and an odd number % 2 is always 1.

- You will use it a lot in your programs!
The Order of Operations

Parenthesis ( )

Negation -

Multiply/ Divide/ Remainder / * %

Add/ Subtract + -
How to Evaluate an Expression

3 + 4 * 4 + 5 * (4 + 3) - 1

3 + 4 * 4 + 5 * 7 - 1

3 + 16 + 5 * 7 - 1

3 + 16 + 35 - 1

19 + 35 - 1

54 - 1

53

(1) inside parentheses first
(2) multiplication
(3) multiplication
(4) addition
(5) addition
(6) subtraction
NOTE

Calculations involving floating-point numbers are approximated because these numbers are not stored with complete accuracy. For example,

\[
\text{System.out.println(1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);}\\
\]
displays 0.5000000000000001, not 0.5, and

\[
\text{System.out.println(1.0 - 0.9);}\\
\]
displays 0.09999999999999998, not 0.1. Integers are stored precisely. Therefore, calculations with integers yield a precise integer result.
public class ComputeArea {
    public static void main(String[] args) {
        //Define inputs and outputs
        //input: radius, output: area
        double radius;
        double area;

        //assign values to inputs and (outputs if necessary)
        radius = 20;

        //calculate area based on radius
        area = radius * radius * 3.14159;

        //show it to user
        System.out.println("The area for the circle of radius " + radius + " is " + area);
    }
}
Reading Input from the Console

1. Let the user know what kind of input you need!

How can we do it in our computeArea program?
Reading Input from the Console

1. Let the user know what kind of input you need!

   How can we do it in our computeArea program?

   System.out.println("Enter a radius!");
Reading Input from the Console

2. Create a Scanner object

```java
Scanner myScanner = new Scanner(System.in);
```

Java keyword for Initializing objects

datatype

variableName
Reading Input from the Console

2. Create a Scanner object

```java
Scanner myScanner = new Scanner(System.in);
```

Java keyword for Initializing objects
Reading Input from the Console

3. Based on the datatype of your input, use an appropriate method.

   radius = myScanner.nextDouble();
Reading Input from the Console

//reading the radius from the terminal/console

System.out.print("Enter a radius: ");

Scanner myScanner = new Scanner(System.in);

radius = input.nextDouble();
Scanner Methods

nextByte(), nextShort(), nextInt(), nextLong(), nextFloat(), nextDouble()

to obtain a byte, short, int, long, float, or double value.
public class ComputeAreaUsingScanner {
    public static void main(String[] args) {
        final double PI = 3.14159;
        //Define inputs and outputs
        double radius;
        double area;

        //read the inputs from the user and (initialize outputs if necessary)
        //Tell the user to enter a radius
        System.out.println("Enter a radius!");
        //Create an Scanner object
        Scanner myScanner = new Scanner(System.in);
        //Read a double value from the scanner and assign it to the radius
        radius = myScanner.nextDouble();

        //calculate area based on radius
        area = radius * radius * PI;

        //show it to user
        System.out.println("The area for the circle of radius " + radius + " is " + area);
    }
}
```java
public class ComputeAreaUsingScanner {
    public static void main(String[] args) {
        final double PI = 3.14159;
        // Define inputs and outputs
        double radius;
        double area;

        // read the inputs from the user and (initialize outputs if necessary)
        // Tell the user to enter a radius
        System.out.println("Enter a radius!");
        // Create an Scanner object
        Scanner myScanner = new Scanner(System.in);
        // Read a double value from the scanner and assign it to the radius
        radius = myScanner.nextDouble();

        // calculate area based on radius
        area = radius * radius * PI;

        // show it to user
        System.out.println("The area for the circle of radius " + radius + " is " + area);
    }
}
```