

# Chapter 2: Primitive Data Types and Operations

## 👉 Objectives

- Write a simple Java program
- The `MyInput` class
- Identifiers, Variables, and Constants
- Primitive Data Types
  - ◆ Byte, short, int, long, float, double, char, boolean
- Operators
  - ◆ +, -, \*, /, %, +=, -=, \*=, /=, %=, ++, --
- Expressions
- Style and Documentation
- Syntax Errors, Runtime Errors, and Logic Errors



# Write a Simple java program

## Example 2.1: Computing the area of a Circle

This program reads the radius from the keyboard and computes the area of the circle. Then display the result on screen.

Math:  $\text{area} = \pi \times \text{radius}^2$

Input and output required



# Write a Simple java program

Structure of a Java file, i.e., ComputeArea

```
public class ComputeArea  
{  
  ...  
}
```

← Data member and methods  
to be given



# Write a Simple java program

Structure of a Java file, i.e., ComputeArea

```
// ComputeArea.java: Compute the area of a circle
public class ComputeArea
{
    public static void main(String[] args)
    {
        ...
    }
}
```


No Data but one methods called  
main() is given



# Write a Simple java program

```
public class ComputeArea
{
// ComputeArea.java: Compute the area of a circle
public class ComputeArea
{
// Main method
public static void main(String[] args)
{
    procedure 1 read in radius
    procedure 2 compute area
    procedure 3 display the area result
}
}
}
```

Need to be implemented



# Question: How we can input data from keyboard?

- The `MyInput` Class
  - This class contains the methods for reading an integer, a double, or a string from the keyboard.
  - The methods are `readInt`, `readDouble`, and `readString`
  - The content of the `MyInput.java`



# Question: How we can display data on screen?

- Use Java built-in class System

```
System.out.println("The area is " + area);
```



```
// MyInput.java: Contain the methods for reading int, double, and
// string values from the keyboard
import java.io.*;

public class MyInput
{
    // Read a string from the keyboard
    public static String readString()
    {
        BufferedReader br
            = new BufferedReader(new InputStreamReader(System.in), 1);

        // Declare and initialize the string
        String string = "";
```





```
// Get the string from the keyboard
try
{
    string = br.readLine();
}
catch (IOException ex)
{
    System.out.println(ex);
}

// Return the string obtained from the keyboard
return string;
}
```



```
// Read an int value from the keyboard
```

```
public static int readInt()
```

```
{
```

```
    return Integer.parseInt(readString());
```

```
}
```

```
// Read a double value from the keyboard
```

```
public static double readDouble()
```

```
{
```

```
    return Double.parseDouble(readString());
```

```
}
```

```
// Read a byte value from the keyboard
```

```
public static byte readByte()
```

```
{
```

```
    return Byte.parseByte(readString());
```

```
}
```



```
// Read a short value from the keyboard
public static short readShort()
{
    return Short.parseShort(readString());
}
// Read a long value from the keyboard
public static long readLong()
{
    return Long.parseLong(readString());
}

// Read a float value from the keyboard
public static float readFloat()
{
    return Float.parseFloat(readString());
}
}
```



# Question: How we can input data from keyboard?

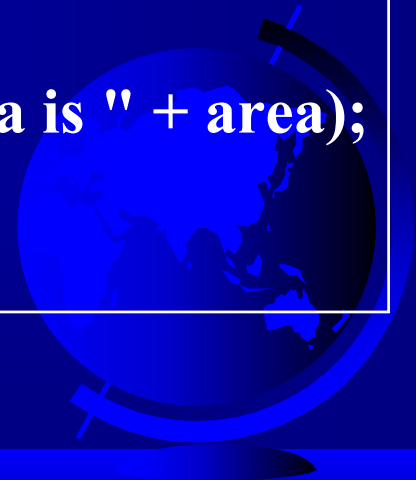
- Alternatively, you can use JOptionPane class in Java swing package, which brings graphic interface for input
- Add `import javax.swing.*;` before the class; then use the following line

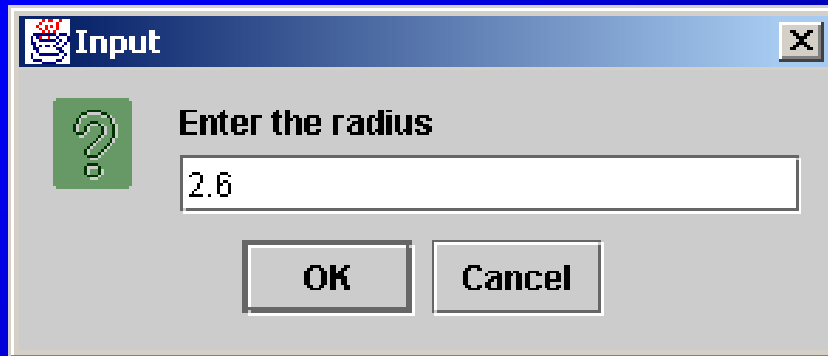
```
JOptionPane.showMessageDialog(null, "The area is " + area);
```



```
import javax.swing.*;

public class ComputeAreaNew
{
    public static void main(String[] args)
    {
        double radius, area;
        final double PI = 3.14159;    // Declare a constant
        radius = Double.parseDouble
            (JOptionPane.showInputDialog("Enter the radius"));
        area = radius*radius*PI;
        JOptionPane.showMessageDialog(null, "The area is " + area);
    }
}
```



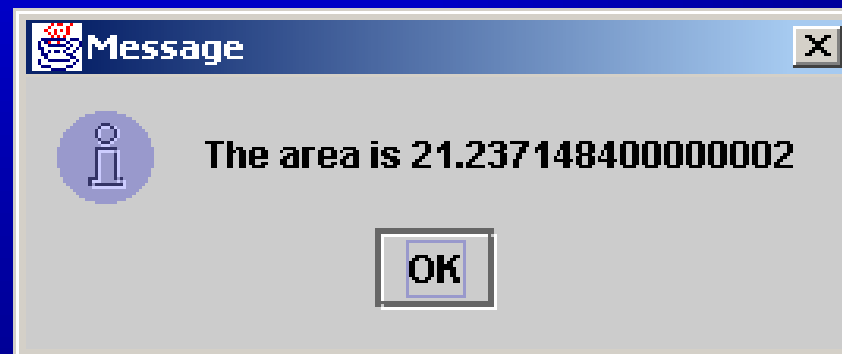
An input dialog box titled "Input" with a close button (X) in the top right corner. It features a green question mark icon on the left. The main text reads "Enter the radius". Below this is a text input field containing the value "2.6". At the bottom, there are two buttons: "OK" and "Cancel".

**Input**

Enter the radius

2.6

OK Cancel

A message dialog box titled "Message" with a close button (X) in the top right corner. It features a blue information icon on the left. The main text reads "The area is 21.237148400000002". At the bottom, there is a single button: "OK".

**Message**

The area is 21.237148400000002

OK



# Identifiers -- Naming


- ☞ Naming variable, constant, method, object, class, files
  - An identifier must start with a letter, an underscore '\_', a dollar sign, '\$', or digitals.
  - An identifier cannot contain operators, such as +, -, and so on.
  - Case-sensitive (like C/C++)
  - Not start with a digital number
  - An identifier cannot be a reserved word.
  - An identifier cannot be `true`, `false`, or `null`.
- ☞ An identifier can be of any length.



# Identifiers -- Naming

## ☞ Reserved keywords

abstract	boolean	break	byte	case
cast	catch	char	class	const
continue	default	do	double	else
extends	final	finally	float	for
future	generic	goto	if	implement
import	inner	instanceof	int	interface
Long	native	new	null	operator
Outer	package	private	protected	public
Rest	return	short	static	super
Switch	synchronized	this	throw	throws
transient	try	var	void	volatile
while				





# Identifiers -- Naming

☞ Naming variable, constant, method, object, class, files

- Underscore, letters, \$, and digitals (Table 2.1)
- Case-sensitive (like C/C++)
- Not start with a number
- No illegal character (operators), such as -, +, \*, <, > ...
- No reserved keywords

☞ Conventional

- All capitals are constant
- First capital is class name
- Variables and objects start with low case



# Variables

```
// Compute the first area  
radius = 1.0;  
area = radius*radius*3.14159;  
System.out.println("The area is " +  
    area + " for radius "+radius);
```

```
// Compute the second area  
radius = 2.0;  
area = radius*radius*3.14159;  
System.out.println("The area is " +  
    area + " for radius "+radius);
```



# Declaring Variables

```
int x;           // Declare x to be an
                 // integer variable;

double radius;  // Declare radius to
                 // be a double variable;

char a;         // Declare a to be a
                 // character variable;
```



# Assignment Statements

```
x = 1;           // Assign 1 to x;  
radius = 1.0;   // Assign 1.0 to radius;  
a = 'A';        // Assign 'A' to a;
```



# Declaring and Initializing in One Step

☞ `int x = 1;`

☞ `double d = 1.4;`

☞ `float f = 1.4;`



# Constants

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

```
final double PI = 3.14159;
```

```
final int SIZE = 3;
```



# Numerical Data Types

byte	8 bits	8-bit signed
short	16 bits	16-bit signed
int	32 bits	32-bit signed
long	64 bits	64-bit signed
float	32 bits	32-bit IEE 755
double	64 bits	64-bit IEEE 754

See Table 2.1 on Page 38



# Number Literals

☞ `int i = 34;`

☞ `long l = 1000000;`

☞ `float f = 100.2f;`

**or**

`float f = 100.2F;`

☞ `double d = 100.2d`

**or**

`double d = 100.2D;`





# Operators

$+$ ,  $-$ ,  $*$ ,  $/$ , and  $\%$

$5/2$  yields an integer 2.

$5.0/2$  yields a double value 2.5

$5 \% 2$  yields 1 (the remainder of the division)



# Shortcut Operators

<i>Operator</i>	<i>Example</i>	<i>Equivalent</i>
<code>+=</code>	<code>i += 8</code>	<code>i = i + 8</code>
<code>-=</code>	<code>f -= 8.0</code>	<code>f = f - 8.0</code>
<code>*=</code>	<code>i *= 8</code>	<code>i = i * 8</code>
<code>/=</code>	<code>i /= 8</code>	<code>i = i / 8</code>
<code>%=</code>	<code>i %= 8</code>	<code>i = i % 8</code>



# Increment and Decrement Operators

➡ `x = 1;`

➡ `y = 1 + x++;`

➡ `y = 1 + ++x;`

➡ `y = 1 + x--;`

➡ `y = 1 + --x;`



# Numeric Type Conversion

Consider the following statements:

```
byte i = 100;
```

```
long l = i*3+4;
```

```
double d = i*3.1+1/2;
```

```
int x = 1; (Wrong)
```

```
long l = x; (fine, implicit  
casting)
```



# Type Casting

➤ double

➤ float

➤ long

➤ int

➤ short

➤ byte



# Type Casting, cont.

Implicit casting

```
double d = 3; (type widening)
```

Explicit casting

```
int i = (int)3.0; (type  
narrowing)
```



# Character Data Type

```
char letter = 'A'; (ASCII)
```

```
char numChar = '4'; (ASCII)
```

```
char letter = '\u000A'; (Unicode)
```

```
char letter = '\u000A'; (Unicode)
```

Special characters

```
char tab = '\t';
```



# Unicode Format

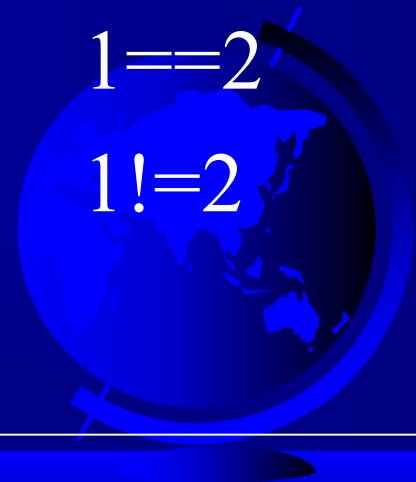
<i>Description</i>	<i>Escape Sequence</i>	<i>Unicode</i>
Backspace	\b	\u0008
Tab	\t	\u0009
Linefeed	\n	\u000a
Carriage return	\r	\u000d
Backslash	\\	\u005c
Single Quote	\'	\u0027
Double Quote	\"	\u0022





# Logical Operators

Operator	meaning	expression
<	less than	$1 < 2$
<=	less than or equal to	$1 <= 2$
>	Greater than	$1 > 2$
>=	greater than or equal to	$1 >= 2$
==	equal to	$1 == 2$
!=	not equal	$1 != 2$



# The boolean Type and Operators

```
boolean lightsOn = true;
```

```
boolean lightsOn = false;
```

Operator	expression
☞ && (and)	(1 < x) && (x < 100)
☞    (or)	(lightsOn)    (isDayTime)
☞ ! (not)	!(isStopped)



# Operator Precedence

☞ Casting

☞ ++, -- (preincrement, predecrement)

☞ \*, /, %

☞ +, -

☞ <, <=, >, >=

☞ ==, !=;

☞ &&, &

☞ ||, |

☞ =, +=, -=, \*=, /=, %=



# Example 2.2

## Computing Mortgages

This program lets the user enter the interest rate, number of years, and loan amount and computes monthly payment and total payment, based on the following formula

$$\frac{\text{loanAmount} \times \text{monthlyInterestRate}}{1 - \frac{1}{(1 + \text{monthlyInterestRate})^{\text{numOfYears} \times 12}}}$$



```
// ComputeMortgage.java: Compute mortgage payments
public class ComputeMortgage
{
    // Main method
    public static void main(String[] args)
    {
        double annualInterestRate;
        int numOfYear;
        double loanAmount;
        // Enter monthly interest rate
        System.out.println(
            "Enter yearly interest rate, for example 8.25: ");
        annualInterestRate = MyInput.readDouble();
        // Obtain monthly interest rate
        double monthlyInterestRate = annualInterestRate/1200;
        // Enter number of years
```



```
System.out.println
```

```
("Enter number of years as an integer, for example 5: ");
```

```
numOfYears = MyInput.readInt();
```

```
// Enter loan amount
```

```
System.out.println("Enter loan amount, for example 120000.95: ");
```

```
loanAmount = MyInput.readDouble();
```

```
// Calculate payment
```

```
double monthlyPayment = loanAmount*monthlyInterestRate/  
    (1 - 1/(Math.pow(1 + monthlyInterestRate, numOfYears*12)));
```

```
double totalPayment = monthlyPayment*numOfYears*12;
```

```
// Display results
```

```
System.out.println("The monthly payment is " + monthlyPayment);
```

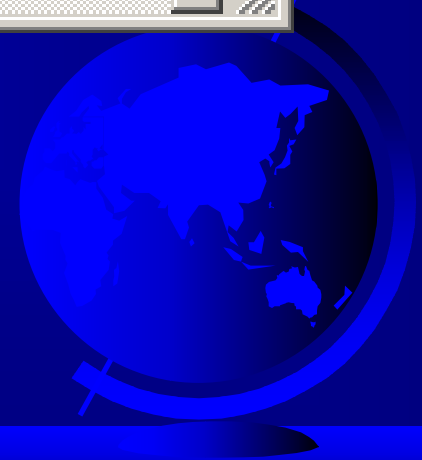
```
System.out.println("The total payment is " + totalPayment);
```

```
}
```

```
}
```



```
C:\WINNT\System32\cmd.exe
Enter yearly interest rate, for example 8.25:
6.5
Enter number of years as an integer, for example 5:
30
Enter loan amount, for example 120000.95:
350000
The monthly payment is 2212.2380822253785
The total payment is 796405.7096011364
Press any key to continue . . .
```



# Example 2.3

## Computing Changes

This program lets the user enter the amount in decimal representing dollars and cents and output a report listing the monetary equivalent in single dollars, quarters, dimes, nickels, and pennies.

Your program should report maximum number of dollars, then the maximum number of quarters, and so on, in this order.





```
// ComputeChange.java: Break down an amount into smaller units
public class ComputeChange
{
    // Main method
    public static void main(String[] args)
    {
        double amount; // Amount entered from the keyboard

        // Receive the amount entered from the keyboard
        System.out.println(
            "Enter an amount in double, for example 11.56");
        amount = MyInput.readDouble();
        int remainingAmount = (int)(amount*100);

        // Find the number of one dollars
        int numOfOneDollars = remainingAmount/100;
        remainingAmount = remainingAmount%100;
```



```
// Find the number of quarters in the remaining amount  
int numOfQuarters = remainingAmount/25;  
remainingAmount = remainingAmount%25;
```

```
// Find the number of dimes in the remaining amount  
int numOfDimes = remainingAmount/10;  
remainingAmount = remainingAmount%10;
```

```
// Find the number of nickels in the remaining amount  
int numOfNickels = remainingAmount/5;  
remainingAmount = remainingAmount%5;
```

```
// Find the number of pennies in the remaining amount  
int numOfPennies = remainingAmount;
```



```
// Display results
```

```
System.out.println("Your amount " + amount + " consists of ");
```

```
System.out.println(numOfOneDollars + "\t dollars");
```

```
System.out.println(numOfQuarters + "\t quarters");
```

```
System.out.println(numOfDimes + "\t dimes");
```

```
System.out.println(numOfNickels + "\t nickels");
```

```
System.out.println(numOfPennies + "\t pennies");
```

```
}
```

```
}
```



```
C:\WINNT\System32\cmd.exe
Enter an amount in double, for example 11.56
23.43
Your amount 23.43 consists of
23      dollars
1       quarters
1       dimes
1       nickels
3       pennies
Press any key to continue . . .
```



# Programming Style and Documentation

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Block Styles



# Appropriate Comments

Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

Include your name, class section, instruction, date, and a brief description at the beginning of the program.



# Naming Conventions

- ☞ Choose meaning 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. For example, the constant `PI`.





# Proper Indentation and Spacing

## ☞ Indentation

- Indent two spaces.

## ☞ Spacing

- Use blank line to separate segments of the code.



# Block Styles

- ☞ Use same-line and next-line styles for braces.

```
public class Account {  
    ...  
}
```

```
public class Account  
{  
    ...  
}
```



# Programming Errors

## ☞ Syntax Errors

- Detected by the compiler

## ☞ Runtime Errors

- Causes the program to abort

## ☞ Logic Errors

- Produces incorrect result

