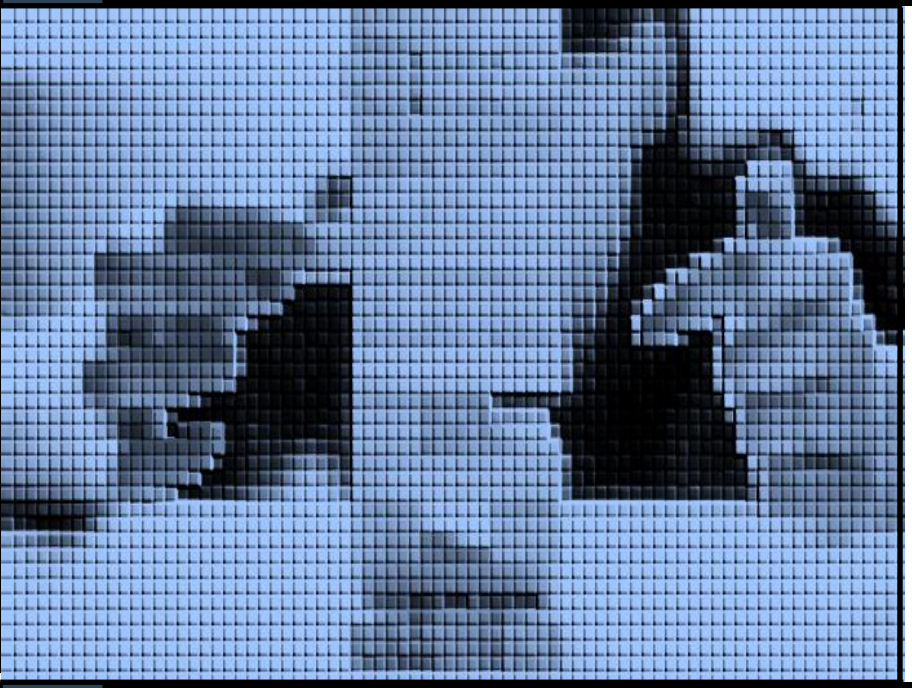
University of BRISTOL

School of Computer Science

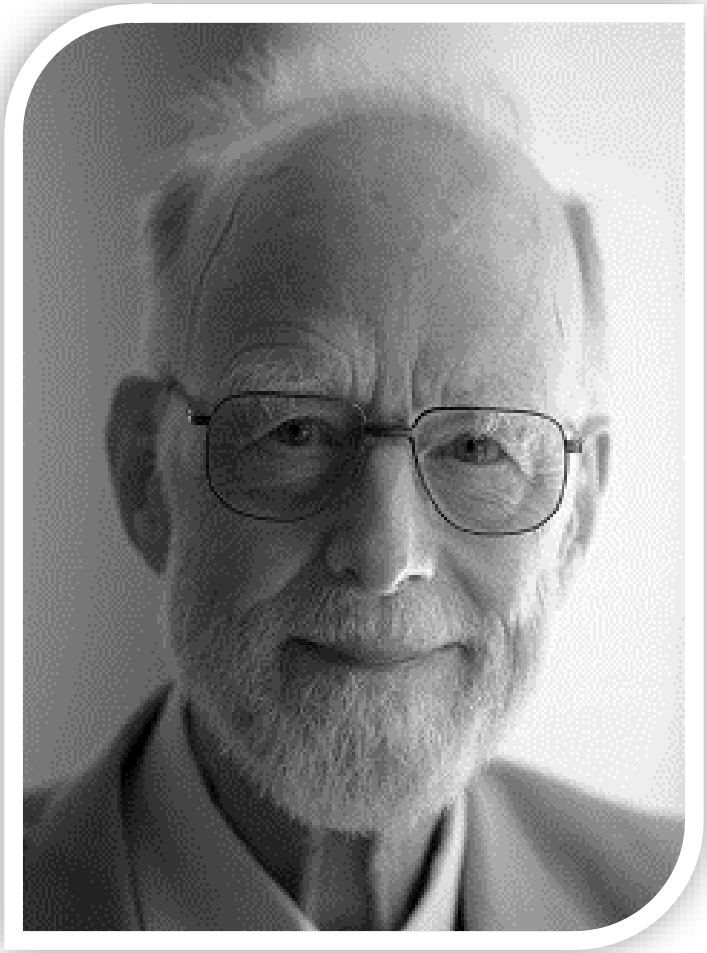
Programming in Java



Programming in Java – Lecture 02

Types, Variables & Scope

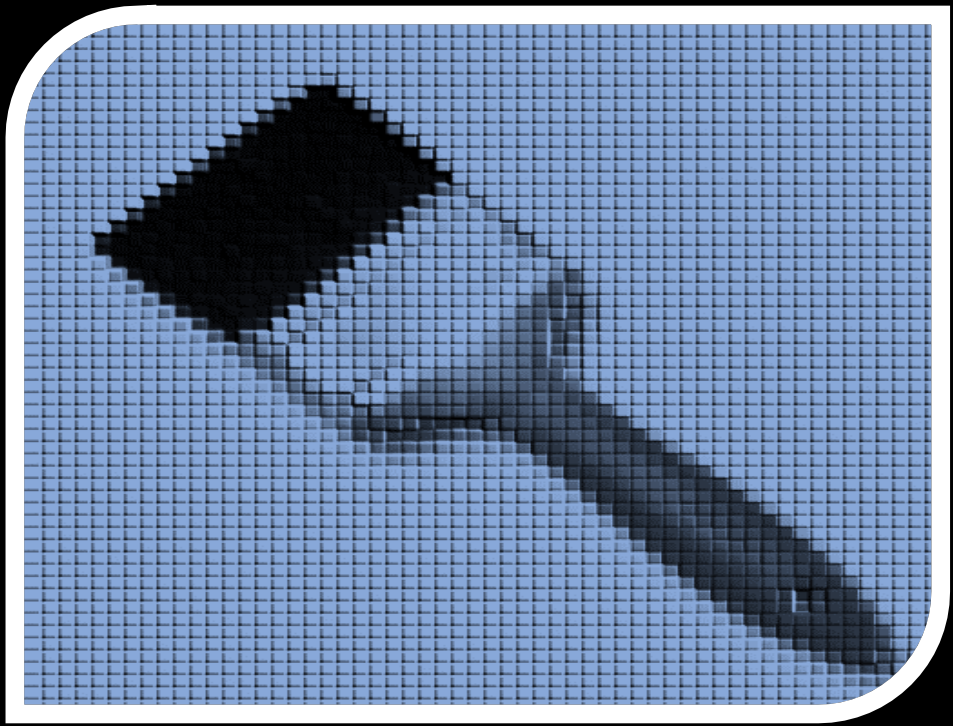
Sion Hannuna and Simon Lock,
based on Tilo Burghardt's C Unit



The most important property of a program is whether it accomplishes the intention of its user.

Tony Hoare (born 1934)
computer scientist

Paint Calculator



How much paint do I need?

```java
public class PaintArea {

    // Calculate area of walls and ceiling.
    public static int area(int length, int width, int height) {
        return 2 * (width + length) * height + length * width;
    }

    // Main method to print out the area of paint for my room.
    public static void main(String[] args) {
        System.out.printf("The paint area is: %d\n", area(5, 3, 2));
    }
}
```

PaintArea.java

Method Call-by-Value

- You call a procedure with multiple arguments by passing it some values in the *correct order*:

```
. . .
public static int area(int length, int width, int height) {
   return 2 * (width + length) * height + length * width;
}

public static void main(String[] args) {
   System.out.printf("The paint area is: %d\n", area(5, 3, 2));
}
```

- In case the procedure `returns` a value (here of type `int`), then the caller can *use* this returned value *in place* of the method call.

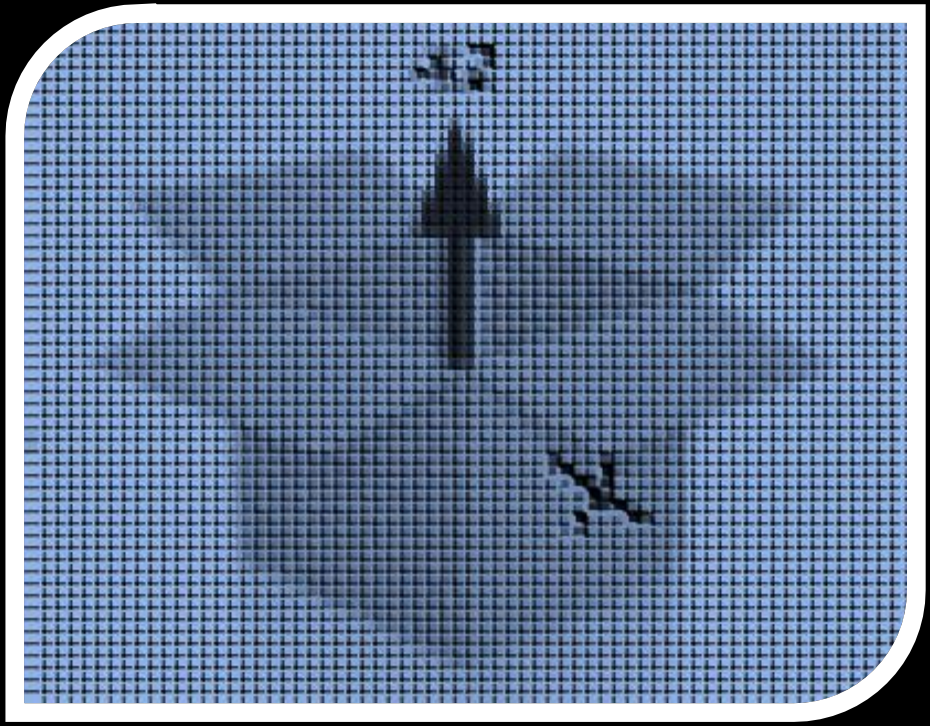
Printing Values

```
. . .
System.out.printf("The paint area is: %d\n", area(5, 3, 2));
. . .
```

replace the %d with the result returned by the getArea function

- The %d is a *conversion specification*. It means 'print an integer in decimal format here' and use the next extra argument to the function as the integer to print.
- The `printf` method can contain many %ds and, thus, many further arguments – such functions are called *variadic*.

Local Variables



Local Variables and Assignments

- An `int` variable in Java is a ‘container' with a *name* and an integer in it:

```
{ ...
  int myVarName = 42;
...
```

- The name is also called the variable’s *identifier*.
- We only need to specify the type `int` the first time we use a variable.
- This first point of use is called the *declaration* of the variable.

- We can re-use a `container’ later in the block and put a different integer in it:

```
...
  myVarName = 43;
...
```

- Or, for instance, copy the contents of another variable into it:

```
...
  myVarName = otherVarName;
...
```

- Giving a value to a variable via ‘=‘ is called an *assignment*.

Easier-to-read Program with Interim Results

```
public class PaintAreaEasier {

   // Calculate area of walls and ceiling.
   public static int area(int length, int width, int height) {
      int sides = 2 * length * height;
      int ends = 2 * width * height;
      int ceiling = length * width;
      return sides + ends + ceiling;
   }

   // Main method to print out the area of paint for my room.
   public static void main(String[] args) {
      int total = area(5, 3, 2);
      System.out.printf("The paint area is: %d\n", total);
   }
}
```

PaintAreaEasier.java

Assignment Statements

- As seen, the '=' operator can be used in an *assignment statement* to set the variable before the operator (e.g. `sides`) `equal to' the value of an entire *expression* that follows :

```
...
  public static int area(int length, int width, int height) {
    int sides = 2 * length * height;
    int ends = 2 * width * height;
    int ceiling = length * width;
    return sides + ends + ceiling;
  }
...
```

- Calculations should be broken up into *small steps* that are easy to understand and to check.
- Note that *all* variables needed for a calculation must be declared and assigned a value *before* they are used.

Scope

- After declaration local variables can only be used *within the current block* (surrounded by { . . . }) they are defined in.
- This valid local environment is known as the variable's *scope*.
- Scope of argument variables is limited to the function body.
- E.g., scope of the variable `sides` is shown in yellow below:

```
…
  public static int area(int length, int width, int height) {
    int sides = 2 * length * height;
    int ends = 2 * width * height;
    int ceiling = length * width;
    return sides + ends + ceiling;
  } // this closing curly brace ends the scope of `sides'
…
  // cannot use `sides' here, it is out-of-scope…
…
```

Expressions

Arithmetic Expressions

- Expressions must be *well-formed* and can use operators, constant values and all *in-scope* arguments and variables:

```
... {
  ... 2 * length * height;
  ... 2 * width * height;
  ... length * width;
  ... sides + ends + ceiling;
} ...
```

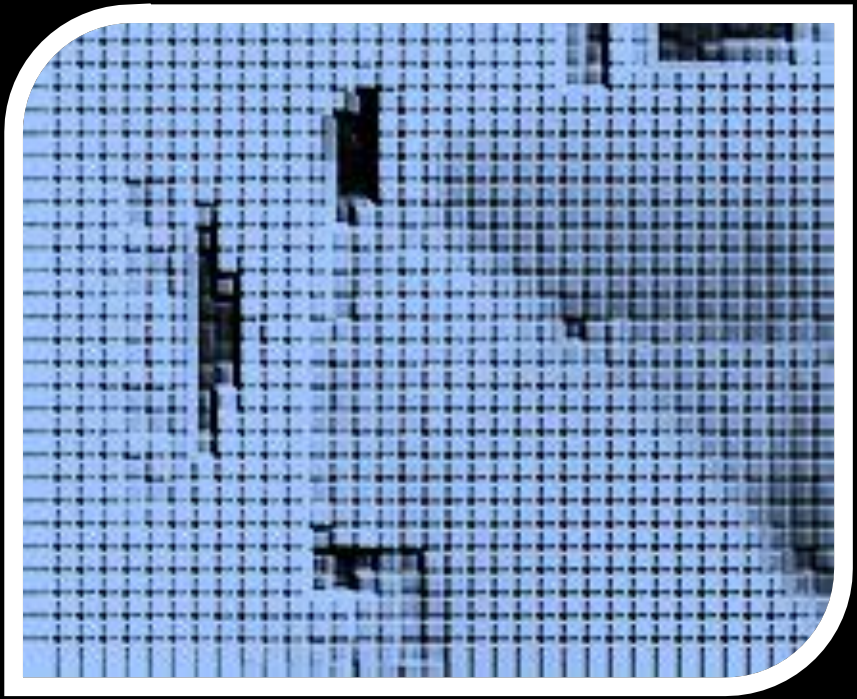
- Parentheses can be used in C expressions as in algebraic expressions, for instance: `a * ( b + c )`
- Place redundant parentheses in expressions to make them clearer if needed.

Arithmetic Operators

- The following arithmetic operators can be used in C:

OPERATION	OPERATOR	ARITHMETIC EXPRESSION	JAVA EXPRESSION	EVALUATION ORDER
Parenthesis	( )	( )	( )	*evaluated first*
Multiplication	*	xy	x * y	*evaluated second, left to right*
Division	/	$x \div y$	x / y	
Remainder	%	$x \bmod y$	x % y	
Addition	+	$x + y$	x + y	*evaluated last, left to right*
Subtraction	-	$x - y$	x - y	

Input



Reading Values from the Keyboard

```
import java.util.Scanner;

public class UserInput {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Please enter an integer: ");
      int length = scanner.nextInt(); // input integer and press return
      . . .
```

UserInput.java

- **Library for Input:**
  - Use import **java.util.Scanner;** to include the Scanner class, which allows you to read user input
- **Create a Scanner Object:**
  - Instantiate a Scanner object with **Scanner scanner = new Scanner(System.in);** to set up for reading input from the console.
- **Declare and Initialize Variable:**
  - Declare an integer variable with **int length;** and use **length = scanner.nextInt();** to read an integer input from the user and store it in the variable.

A Complete Paint Area Calculator

```java
import java.util.Scanner;

public class PaintAreaCalculator {

  // Calculate area of walls and ceiling.
  public static int area(int length, int width, int height) {
    int sides = 2 * length * height;
    int ends = 2 * width * height;
    int ceiling = length * width;
    return sides + ends + ceiling;
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter room length:");
    int length = scanner.nextInt();
    System.out.println("Enter room width:");
    int width = scanner.nextInt();
    System.out.println("Enter room height:");
    int height = scanner.nextInt();
    int total = area(length, width, height);
    System.out.println("The paint area is: " + total);
    scanner.close();
  }
}
```

`PaintAreaCaclulator.java`

Use commas ‘`,`‘ to declare *multiple variables* of the same type in a single line.

Note that local variables in the `main` procedure and the arguments of the function `area` have the same names! – However, note that they have disjunct scopes and, thus, no *name clash* occurs.

Remember to *declare-and-initialise* variables (such as `total`) in one statement whenever possible without producing *dead code*.