

Lecture 3

Class Methods and Arrays

Student Representatives

- Selma Moqvist (mselma@student.chalmers.se)
- Valter Schütz (schutzv@student.chalmers.se)

Exercise Sessions? What?

- Doing exercises from the book
 - ...and getting help when you're stuck
 - Pen & paper preferred (it'll help you on the exam)
- Getting answers about the course material
- At the end: teacher solving an exercise or two on the board
- Solving a full exam before exam date!

Extended Example

Guess the Number Game

The user picks a random number between 1 and 100. The computer must guess the number.

After every wrong guess, the user tells the user “higher” or “lower”.

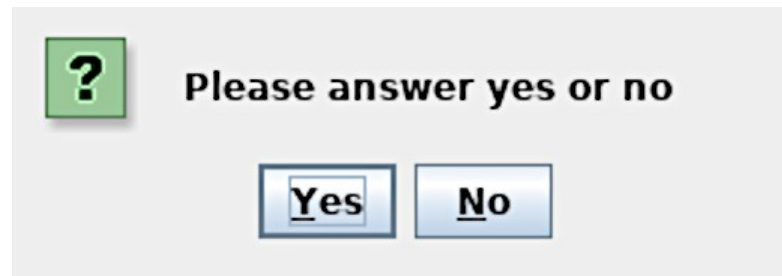
The user wins if the computer find guess the number within 5 guesses.

Extended Example

Guess the Number Game

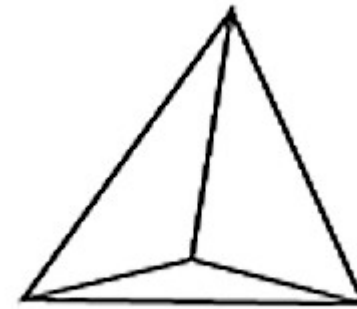
```
int result =  
    showConfirmDialog(  
        null,  
        "Please answer yes or no",  
        "Title of the window",  
        YES_NO_OPTION  
    );
```

- `result == 0` if yes, `1` if no



Example: a simple 3D renderer

- Drawing simple 3D shapes onto a window
- Our tools:
 - Loops
 - Arrays
 - Class methods
 - A bit of vector and matrix math



Example: some 3D vector math

Class Methods

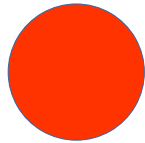
A class method (function, procedure) is defined by

```
static return-type name (type1 arg1, type2 arg2, ...) {  
    block  
    return value;  
}
```

```
static void name (type1 arg1, type2 arg2, ...) {  
    block  
}
```

We can make them `public` (visible to other classes) or `private` (visible only inside this class).

Variable Declaration and Assignment



Syntax

Declaration: *type varName, ..., varName ;*

Assignment: *varName = expression ;*

Initialization: *type varName = expression ;*

(Types seen so far: `String`, `int`, `double`)

Java is a **strongly typed** language:

A variable must have one and only one type.

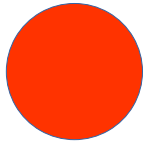
Java is a **statically typed** language:

A variable cannot change its type.

These restrictions make it harder to write code.

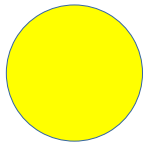
This is a good thing – they make it harder to write incorrect code.

Variable Declaration



If variables are declared inside a block, then they are destroyed at the end of the block.

The block is called the *scope* of the variable.



Declare variables at the start of a block

Keep the scope of a variable as small as possible

One variable declaration per line

Align types and variables names:

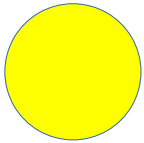
```
int      numCustomers;
```

```
int      numProducts;
```

```
double  profit;
```

Initialize variables if possible

Naming Variables



Use meaningful names

When I read your code, I should understand what each variable's purpose is

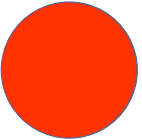
Avoid single letters (except for loop variables – see later)

Avoid very similar variable names

Put some time and thought into naming your variables!

Comments

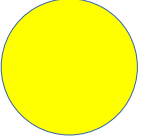
Comments are text for human readers. They are ignored by the compiler.



```
// Comment lasts until end of line
```

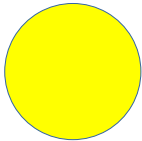


```
/* Comment runs over several lines  
   and lasts until  
   the symbols */
```



Comment every method to explain its purpose, the meaning of each argument, and the return value.

Comments



- Don't explain the obvious:

```
// Increase balance by 0.03 of its amount  
balance += balance * 0.03;
```

The code should say **what** you are doing.
The comments explain **why**.

```
// Apply interest rate of 3%  
balance += balance * 0.03
```

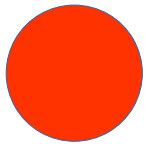
- Beware of “comments as deodorant”

Bad: Leave code messy and write a comment explaining what how you should have written it.

Good: Clean up the code

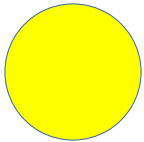
Whitespace

Spaces, tabs, newlines are called **whitespace**.



Syntax

Whitespace is ignored by the compiler



Best Practice

Use indentation to show where blocks begin and end.

When starting a new statement, start a new line

Two Ways to Use Braces

Egyptian braces:

```
if (a == 2) {  
    b = 1;  
}
```


```
if (a == 2)  
{  
    b = 1;  
}
```



Choose one of these styles and use it consistently

Watch out for “Hadouken Code”!

```
public static void registerUser(User user) {  
    if(user.getName() != null) {  
        if(user.getName() != "") {  
            if(user.getPassword() != null) {  
                if(user.getPassword().length() > 5) {  
                    if(user.getConfirmPassword() != null) {  
                        if(user.getConfirmPassword() == user.getPassword()) {  
                            if(user.getName() != null) {  
                                if(isValidName(user.getName())) {  
                                    if(user.getEmail() != null) {  
                                        if(isValidEmail(user.getEmail())) {  
                                            if(user.getConfirmEmail() != null) {  
                                                if(user.getEmail() == user.getConfirmEmail()) {  
                                                    createUser(user);  
                                                    setResponse(200, "User registration successful!");  
                                                    return;  
                                                }  
                                            }  
                                        }  
                                    }  
                                }  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
    setResponse(500, "Registration failed!");  
}
```



If indentation gets too deep, consider refactoring your code. Introduce a method, or fewer `if`s with larger expressions.

Example: more 3D vector math

Arrays

An **array** is an object that holds several values. The size is fixed, and all the values must have the same type.

```
int[] a = new int[3];  
a[0] = 2;  
a[1] = 3;  
a[2] = 4;
```

or:

```
int[] a = {2, 3, 4};
```

Arrays

```
int[] a = new int[10];
```

creates an array of size 10 that holds integers.

These 10 integer values – the **elements** of a - are referred to by:

```
a[0], a[1], a[2], a[3], a[4], a[5],  
a[6], a[7], a[8], a[9]
```

5 is the **index** of a[5]

Arrays

- Reading the value of an element:
`System.out.println("The answer is: " + a[5]);`
- Changing the value of an element:
`a[7] = 23;`
`a[n] = 15;`
`a[n+1] = a[n] + a[n-1];`

Arrays

Arrays become powerful when used with for-loops

Example: Sum of all the elements in an array

```
int sum = 0;
for (int i=0; i<a.length; i++) {
    sum += a[i];
}
```

or

```
int sum = 0;
for (int elem: a) {
    sum += elem;
}
```

N-dimensional arrays

- Arrays can have more than one dimension - “arrays of arrays”

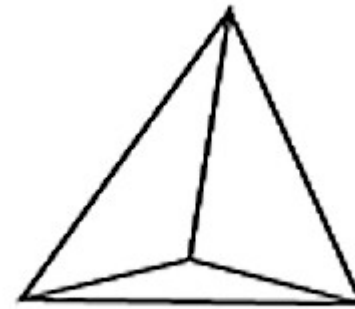
```
int[][] a = new int[2][2];  
a[0][0] = 1;  
a[0][1] = 2;  
a[1][0] = 3;  
a[1][1] = 4;
```

or

```
int[][] a = {{1, 2}, {3, 4}};
```

Example: a simple 3D renderer

- Drawing simple 3D shapes onto a window
- Our tools:
 - Loops
 - Arrays
 - Class methods
 - A bit of vector and matrix math



Reading this week:

- *Java Direkt med Swing* 1.9-1.14, 2.4, 5.2
- *Code Complete* Chapter 8

Exercises this week:

- *Java Direkt med Swing* section 1.16