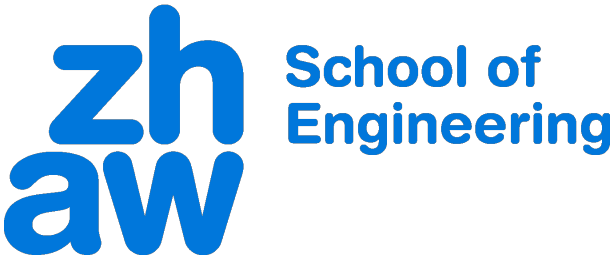
ZHAW Zurich University of Applied Sciences
Winterthur



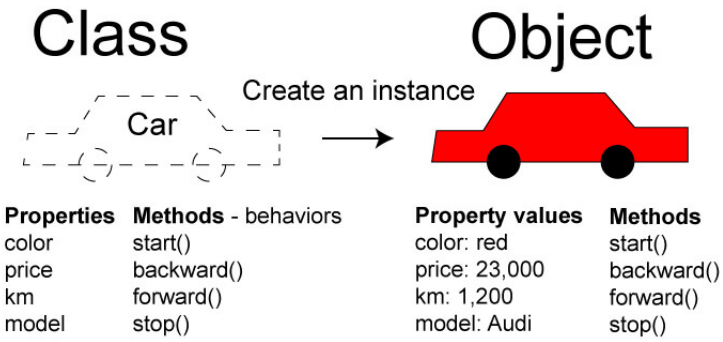
Zusammenfassung INF2
Studienwochen 6-10

Written by: Severin Sprenger
October 13, 2025
Zf. INF2 SW 6-10

1 What is Java?

Java is a cross-platform object oriented programming language. The language is cross-platform because of the use of the JVM (Java Virtual Machine). When Java is compiled during development, it is compiled into byte code that can be executed on the JVM.

2 What is a object?



3 The syntax

The syntax is the same of all basic building blocks, like conditionals and loops.With one difference: The condition passed to a conditional needs to be of the type **boolean**.

3.1 File names

Very important! The class defined in a file has to have the exactly same name is the file it is defined in.

3.2 Datatype

Type	byte	char	short	int	long	float	double	boolean	String
Bits	8	8	16	32	64	32	64	1	

Important! String isn't terminated with a `\0` like in C, can be concatenated using `+` and have a method `.length()` to get the strings length.

3.3 Conversion

- Casting to types like in C
- Parse using for eg. `Integer.parseInt(String)`

3.4 String methods

```
str1.equals(str2)
str1.equalsIgnoreCase(str2)
str1.split(",")
str1.substring(4, 7)
str1.toLowerCase()
str1.toUpperCase()
str1.replace("foo", "bar")
str1.startsWith("gugus")
```

3.5 Class structure

```
public class SomeRandomClass {
    /** Any object properties */

    public SomeRandomClass() {
        /** This is the default constructor */
    }

    /** Some object methods */
}
```

4 Basic functions

4.1 Main function

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

4.2 Constructor

The constructor is a "method" on a class that is called when a new instance of that class is created. The constructor needs to have the same name as the class itself and has no return value. A class can have multiple constructors with different types of arguments.

```
public myTestClass () {}
```

4.3 Print to CLI

```
System.out.print(String) System.out.println(String)
```

4.4 Read from CLI

```
Scanner scanner = new Scanner(System.in);
double foo = scanner.nextDouble();
int bar = scanner.nextInt();
```

4.5 Math class

```
Math.E
Math.PI
double Math.ceil(double x)
double Math.floor(double x)
int Math.round(float x)
double Math.random()
```

4.6 Date class

```
Date now = new Date();
String nowStr = now.toString();
```

4.7 ArrayList class

```
import java.util.ArrayList;
ArrayList<String> gugus = new ArrayList<String>();
gugus.add("foo"); gugus.add("bar");
```

4.8 Arrays

Declaration: `int[] foo;`
Definition: `foo = new Int[8];`
Initialization on declaration: `int[] bar = {1, 2, 3, 4};`
Access: `int gugus = bar[0]; // Important!` Can't read out of bounds in Java.

4.9 Final

A variable that is defined `final` can't be reassigned.

4.10 Static

For executing a static method of a class, no instance is needed and static functions can't modify data on the class.

4.11 This

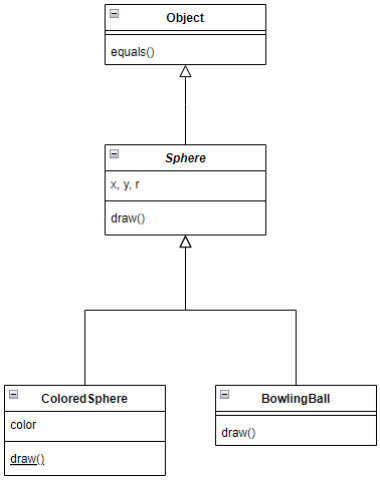
`this` is a reference to the current object and can be used to assign variables on the object.

```
this.gugus = gugus;
```

5 Inheritance

A new class can inherit properties and methods from another "template" class. To use a class as a "template" for another class the following syntax is needed. (Note: All classes in Java inherit from a class named Object)

```
public class SomeNewClass extends YourTemplateClass {
    /** Add constructors, methods and properties to the new class */
}
```



5.1 Variable visibility

Modifier	Same class	Sub class	Other classes
`private`	yes	no	no
`protected`	yes	yes	no
`public`	yes	yes	yes

5.2 Constructors

The class that inherits from a "template" class can have its own constructor, in this constructor the constructor of the "template" class can be called using `super(...)`. If the constructor of the "template" class doesn't require any parameters, the constructor will be called automatically.

6 Casting

In Java, class casting is the process of treating an object of one type as if it were another type. This is commonly used when dealing with inheritance, where objects of subclasses can be treated as objects of their superclass (upcasting) or vice versa (downcasting).

- **Upcasting:** This is the process of casting a subclass object to a superclass reference. It's always safe and doesn't require an explicit cast.
- **Downcasting:** This is the process of casting a superclass reference to a subclass object. It requires an explicit cast and can lead to a ClassCastException if the object is not actually an instance of the subclass.

7 Events (Java GUI's)

In java events are handele in a event loop. If an events was triggered, its event handler (developer implemented) is called.

You can add a event listener using the following syntax. The following code snipped registers the current class as the event handler.

```
SomeSourceObject.addActionListener(this);
```

If an event is triaged by a source object, the method `actionPerformed` is called and the source object is passed as a argument.

```
public void actionPerformed(ActionEvent evt) {
    /** Handle events */
    repaint();
}
```

8 Basic GUI code syntax

```
public class SomeGuiClass extends JFrame implements ActionListener {
    public static void main(String[] args) {
        // Set look and feel from system look and feel
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (Exception e) {
            return;
        }

        SomeGuiClass window = new SomeGuiClass();

        // Set window attributes
        window.setTitle("Some window title");
        window.setSize(WINDOW_WIDTH, WINDOW_HEIGHT);

        // Initialize all components and make the window visible
        window.initComponents();
        window.setVisible(true);
    }

    private void initComponents() {
        JPanel panel = (JPanel) this.getContentPane();

        // Set the layout to be used
        panel.setLayout(new FlowLayout());
    }


    private void initComponents() {}

    public void actionPerformed(ActionEvent e) {}
}
```