Java

Tips

Hannes Ueck, Jakob Krude

17. Dezember 2020

Java-Kurs

Overview

null-value

null-Value

If a primitive variable has been declared but not initialized, its value is 0.

```
int a;
if (a == 0) {
    System.out.println("a is 0");
// prints: a is 0
}
```

For non-primitive types the same concept applies with null.

```
Object object;
if (object == null) {
    System.out.println("object is null");
// prints: object is null
}
```

Problems with float-values

float-value

```
double f1 = 1.1;
double f2 = .0;
for (int i = 1; i <= 11; i++) {
    f2 += .1;
}
System.out.println("f1 = " + f1);
System.out.println("f2 = " + f2);

if (f1 == f2)
    System.out.println("f1 and f2 are equal\n");
else
    System.out.println("f1 and f2 are not equal\n");
}
//Output:
f1 = 1.1
f2 = 1.0999999999999999
f1 and f2 are not equal
}
```

float-values

Float values can differ because of rounding errors. Therefore, you should not compare them with ==.

You can use BigDecimal for mathematical calculations.

More info: `https://howtodoinjava.com/java-examples/correctly-compare-float-double/`

Scopes

Visibilities

```
    class MyGreatClass {

        //Attributes are public by default
        Car myCar;

        //Public are available in every part of our code.
        public Cat myCat;

        //Private Attributes can only be acced via a method
        private House myHouse;
    }


```

If

```
    ...
    int a = 1;
    if(...) {
        int b = 5;
        System.out.println(a);
        System.out.println(b);
    }

    System.out.println(a);
    System.out.println(b);
    ...

```

b is only available in the scope of the if.

b is not outside the if available.

WILL NOT COMPILE

For

```java
    for(int i = 0; i <= 100; i++) {
        int b = 3;
        System.out.println(i);
        System.out.println(b);
    }

```

b will be redefined in every round of the loop and is only available in the for loop.

The scope is created at the beginning and destroyed at the end of each round.

While

```java
    int i = 0;
    while(i <= 100) {
        int b = 3;
        System.out.println(i);
        System.out.println(b);
    }

```

For and while got the same scope behavior.

Examples

```java
    public class myClass {
        private int a;

        public myClass(int a) {
            this.a = a;
        }
    }

```

Use the nearest definition.

In one scope every variable name can be defined only one time.

What we learned

Scopes are definition areas for variables. Every Block defines a new Scope.