Inhaltsverzeichnis

1. Java Loops
   1.1. Preparation
   1.2. Theorie & Einführung
      1.2.1. `for (each)`
      1.2.2. `(do) while`
   1.3. Demonstrationen
   1.4. Exercises
   1.5. References

1. Java Loops

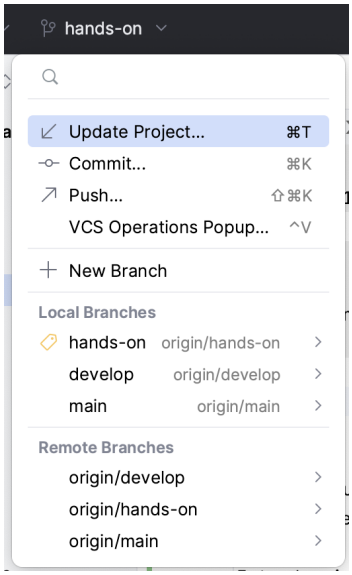
1.1. Preparation

Das **Projekt** bzw. der "*lokale Workspace*", d.h. euer lokales Arbeitsverzeichnis, in dem alle Sourcen liegen, muss als allererstes zum Start in den Tag aktualisiert werden, d.h. ...

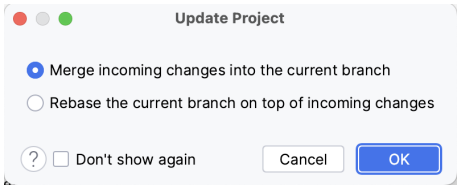
↙ `Update Project…`

Das geht am besten mithilfe der IDE im Menü oder über das GIT Icon:

Das geht am besten mithilfe der IDE im Menü oder über das GIT Icon:



Danach muss - im sich öffnenden Dialog - noch folgendes bestätigt werden: *merge incoming changes into the current branch*



Der Vorgang sollte mit einer Erfolgsmeldung abschließen.

1.2. Theorie & Einführung

1.2.1. `for (each)`

A for loop is a control structure that allows to repeat certain operations by incrementing and evaluating a loop counter.

Before the first iteration, the loop counter gets initialized, then the condition evaluation is performed followed by the step definition (usually a simple incrementation).

The **syntax** of the for loop is:

```java
for (Initialization; Boolean-Expression; Step) {
   statement;
}
```

Let's see a simple example:

```java
for (int i = 0; i < 5; i++) {
   System.out.println("simple for loop: i = " + i);
}
```

The initialization, Boolean-expression, and step used in for statements are optional. Here's an example of an infinite for loop:

```java
for ( ; ; ) {
    // infinite for loop
}
```

Enhanced for Loop

Since Java 5, there is a second kind of for loop called the **enhanced for** which makes it easier to iterate over all elements in an array or a collection.

The syntax of the enhanced for loop is:

```java
for(Type item : items) {
    statement;
}
```

Since this loop is simplified in comparison to the standard for loop, we need to declare only two things when initializing a loop:

1. The **handle** for an element we're currently iterating over
2. The **source** array/collection we're iterating

That means, for each element in items, assign the element to the item variable and run the body of the loop.

Enhanced Loop example:

```java
int[] intArr = { 0,1,2,3,4 };

for (int num : intArr) {
    System.out.println(
        "Enhanced for-each loop: i = " + num);
}
```

We can use it to iterate over various Java data structures:

Given a `List<String> list` object – we can iterate by:

```java
for (String item : list) {
    System.out.println(item);
}
```

Loop with Lamdba (functional interface)

```java
List<String> letters = new ArrayList<>();

letters.add("A");
letters.add("B");
letters.add("C");

letters.forEach(letter -> System.out.println(letter));
```

1.2.2. `(do) while`

while

The while loop is one of Java's fundamental loop statements. It repeats a statement or a block of statements while its controlling Boolean-expression is true.

The **syntax** of the while loop is:

```java
while (Boolean-expression) {
    statement;
}
```

The loop's Boolean-expression is evaluated before the first iteration of the loop – which means that if the condition is evaluated to false, the loop *might not* run even once.

A simple example:

```java
int i = 0;
while (i < 5) {
    System.out.println("While loop: i = " + i++);
}
```

do-while

The do-while loop works just like the while loop except for the fact that the first condition evaluation happens after the first iteration of the loop:

```java
do {
   statement;
} while (Boolean-expression);
```

A simple example:

```java
int i = 0;
do {
   System.out.println("Do-While loop: i = " + i++);
} while (i < 5);
```

1.3. Demonstrationen

Die Unit-Tests zur **Demonstration** finden sich hier:

```
src/test/java/de/dhbw/loops/LoopTests.java
```

Der zugehörige, in den Tests genutzte **Quellcode** findet sich hier:

```
src/main/java/de/dhbw/loops/demo/*.java
```

1.4. Exercises

Nutze folgendes Package für deine **Unit-Tests**:

```
src/test/java/de/dhbw/loops/ExerciseTests.java
```

Die im Test benutzten **Implementierungen** gehören in das Package:

```
src/main/java/de/dhbw/loops/exercises/*.java
```

Übung 1:

Gegeben eine Liste mit Zahlen. Finde den größten Zahlenwert dieser Liste mithilfe eine (alten) `for` (IntelliJ `fori`) Schleife und teste das Ergebnis mithilfe von `assertEquals`.

Optional: Wandele die for Schleife in eine "enhanced for" Schleife um.

Übung 2:

Gegeben ein Startwert von `10`. Implementiere einen *CountDown* mithilfe einer `while` Schleife, die den CountDown solange runterzählt, bis die `0` erreicht ist und dann abbricht.

1.5. References

See also:

- For Loop (https://www.baeldung.com/java-for-loop)
- While Loop (https://www.baeldung.com/java-while-loop)