合肥學院
HEFEI UNIVERSITY

OOP with Java
8. Loops

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

Outline

website

Loops

- Sometimes, we need to carry out the same command again and again in a program.

Loops

- Sometimes, we need to carry out the same command again and again in a program.
- Instead of copying the same code multiple times, we may wrap it into a *loop*.

Repetitive Code

Listing: Vertical Ball Throw: $t = 0, 0.2, 0.4, \ldots, 2.2$, and $2.2s$.

```
/**
 * A ball is thrown vertically upwards into the air by a 1.8m tall person<br/>
 * with velocity 10m/s. Where is it after t=0,0.2,...,2.2 seconds?<br/>
 * x(t) = x0 + v0 * t - 0.5 * g * t^2
 */
public class VerticalBallThrowMultiple {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    double x0 = 1.8d; // initial vertical position
    double v0 = 10d; // initial velocity upwards
    double g = 9.80665d; // free fall acceleration downwards

    double t = 0.0d; // set the time to 0s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 1.8
    t = 0.2d; // set the time to 0.2s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 3.6038669999999997
    t = 0.4d; // set the time to 0.4s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 5.015468
    t = 0.6d; // set the time to 0.6s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 6.034803
    t = 0.8d; // set the time to 0.8s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 6.661872000000001
    t = 1.0d; // set the time to 1.0s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 6.896675000000001
    t = 1.2d; // set the time to 1.2s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 6.739212000000001
    t = 1.4d; // set the time to 1.4s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 6.189483000000003
    t = 1.6d; // set the time to 1.6s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 5.247488000000001
    t = 1.8d; // set the time to 1.8s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 3.913227000000001
    t = 2.0d; // set the time to 2.0s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 2.186700000000002
    t = 2.2d; // set the time to 2.2s
    System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints 0.06790699999999816
  }
}
```

For-Loop

- The most basic loop in Java is a *for loop*

Listing: The structure of *for loop*

```
for( [counter variable initialization]; [loop condition]; [counter increment]) {
   // loop body
}
```

- The (optional) first element `[counter variable initialization]` can be used to allocate and initialize a counter variable. It is executed *before* the loop begins.
- The (optional) second element `[loop condition]` is an expression which is checked before each loop iteration. Only if it is `true`, the `loop body` is executed.
- If there is no loop condition, the loop will loop for ever.
- The (optional) third element `counter increment` can be used to increase or decrease the loop counter. It is executed after each loop iteration.
- The `loop body` may contain any command or statement, including conditionals and other loops.

For-Loop Example 1

Listing: Vertical Ball Throw: $t = 0, 0.2, 0.4, \ldots, 2.2$, and $2.2s$.

```
/**
 * A ball is thrown vertically upwards into the air by a 1.8m tall person<br/>
 * with velocity 10m/s. Where is it after t=0,0.2,...,2.2 seconds?<br/>
 * x(t) = x0 + v0 * t - 0.5 * g * t^2
 */
public class VerticalBallThrowForLoop {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    double x0 = 1.8d; // initial vertical position
    double v0 = 10d; // initial velocity upwards
    double g = 9.80665d; // free fall acceleration downwards

    for (double t = 0.0d; t <= 2.2d; t += 0.2d) { // using double counter: imprecision accumulates
      System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints the current position
    }
  }
}
```

For-Loop Example 2

Listing: Vertical Ball Throw: $t = 0, 0.2, 0.4, \ldots, 2.2$, and $2.2s$.

```
/**
 * A ball is thrown vertically upwards into the air by a 1.8m tall person<br/>
 * with velocity 10m/s. Where is it after t=0,0.2,...,2.2 seconds?<br/>
 * x(t) = x0 + v0 * t - 0.5 * g * t^2
 */
public class VerticalBallThrowForLoopBetter1 {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    double x0 = 1.8d; // initial vertical position
    double v0 = 10d; // initial velocity upwards
    double g = 9.80665d; // free fall acceleration downwards

    for (int i = 0; i < 12; i++) { // using an integer for counting
      double t = 0.2d * i; // and multiplying it by 0.2 is more accurate
      System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints the current position
    }
  }
}
```

For-Loop Example 2

Listing: Vertical Ball Throw: $t = 0, 0.2, 0.4, \ldots, 2.2$, and $2.2s$.

```
/**
 * A ball is thrown vertically upwards into the air by a 1.8m tall person<br/>
 * with velocity 10m/s. Where is it after t=0,0.2,...,2.2 seconds?<br/>
 * x(t) = x0 + v0 * t - 0.5 * g * t^2
 */
public class VerticalBallThrowForLoopBetter2 {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    double x0 = 1.8d; // initial vertical position
    double v0 = 10d; // initial velocity upwards
    double g = 9.80665d; // free fall acceleration downwards

    for (int i = -1; ++i < 12; ) { // using an integer for counting
      double t = 0.2d * i; // and multiplying it by 0.2 is more accurate
      System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints the current position
    }
  }
}
```

Break and Continue

- There are two more keywords that can be used to control loop

Break and Continue

- There are two more keywords that can be used to control loop:
  - `break` exits the current loop immediately

Break and Continue

- There are two more keywords that can be used to control loop:
  - `break` exits the current loop immediately
  - `continue` jumps to the next iteration (executing the incremental and condition checking code of the loop header, if any)

Break and Continue

- There are two more keywords that can be used to control loop:
  - `break` exits the current loop immediately
  - `continue` jumps to the next iteration (executing the incremental and condition checking code of the loop header, if any)
- Normally, we would avoid using these keywords, because they make the code harder to read...

For-Loop Example with Break

Listing: Vertical Ball Throw: $t = 0, 0.2, 0.4, \ldots, 2.2$, and $2.2s$.

```
/**
 * A ball is thrown vertically upwards into the air by a 1.8m tall person<br/>
 * with velocity 10m/s. Where is it after t=0,0.2,...,2.2 seconds?<br/>
 * x(t) = x0 + v0 * t - 0.5 * g * t^2
 */
public class VerticalBallThrowForLoopStrange1 {

  /** The main routine
   * @param args
   *           we ignore this parameter for now */
  public static final void main(String[] args) {
    double x0 = 1.8d; // initial vertical position
    double v0 = 10d; // initial velocity upwards
    double g = 9.80665d; // free fall acceleration downwards

    for (int i = 0;;) { // using an integer for counting, but no loop condition
      double t = 0.2d * i; // and multiplying it by 0.2 is more accurate
      System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints the current position
      if (++i >= 12) { // the actual loop condition
        break; // exit the for loop if above condition is met
      }
    }
  }
}
```

For-Loop Example with Continue and Break

Listing: Vertical Ball Throw: $t = 0, 0.2, 0.4, \ldots, 2.2$, and $2.2s$.

```
/**
 * A ball is thrown vertically upwards into the air by a 1.8m tall person<br/>
 * with velocity 10m/s. Where is it after t=0,0.2,...,2.2 seconds?<br/>
 * x(t) = x0 + v0 * t - 0.5 * g * t^2
 */
public class VerticalBallThrowForLoopStrange2 {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    double x0 = 1.8d; // initial vertical position
    double v0 = 10d; // initial velocity upwards
    double g = 9.80665d; // free fall acceleration downwards

    for (int i = 0;;) { // using an integer for counting, but no loop condition
      double t = 0.2d * i; // and multiplying it by 0.2 is more accurate
      System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints the current position
      if (++i < 12) { // this is the continuation condition
        continue; // jump to next loop iteration
      }
      break; // exit the for loop: happens if "continue" was not reached above
    }
  }
}
```

While Loop

- `for` loops are usually used along with counter variables

While Loop

- `for` loops are usually used along with counter variables
- `while (condition)` loops are loops with any type of (Boolean) `condition` expression

While Loop

- `for` loops are usually used along with counter variables
- `while (condition)` loops are loops with any type of (Boolean) `condition` expression
- The `condition` is checked before every iteration

While Loop

- `for` loops are usually used along with counter variables
- `while (condition)` loops are loops with any type of (Boolean) `condition` expression
- The `condition` is checked before every iteration
- (notice, however, that `while (condition){ ... }` is basically equivalent to `for(; condition; ){ ... }` ... but it's kind of ugly to use `for` loops like this...)

While-Loop Example 1

Listing: Vertical Ball Throw: $t = 0, 0.2, 0.4, \ldots, 2.2$, and $2.2s$.

```
/**
 * A ball is thrown vertically upwards into the air by a 1.8m tall person<br/>
 * with velocity 10m/s. Where is it after t=0,0.2,...,2.2 seconds?<br/>
 * x(t) = x_0 + v_0 * t - 0.5 * g * t^2
 */
public class VerticalBallThrowWhileLoop1 {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    double x0 = 1.8d; // initial vertical position
    double v0 = 10d; // initial velocity upwards
    double g = 9.80665d; // free fall acceleration downwards

    int i = 0;
    while (i < 12) { // the condition is checked at the top of the loop
      double t = 0.2d * i;
      System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints the current position
      i++; // increase counter
    }
  }
}
```

While-Loop Example 2

Listing: Vertical Ball Throw: $t = 0, 0.2, 0.4, \ldots, 2.2$, and $2.2s$.

```java
/**
 * A ball is thrown vertically upwards into the air by a 1.8m tall person<br/>
 * with velocity 10m/s. Where is it after t=0,0.2,...,2.2 seconds?<br/>
 * x(t) = x0 + v0 * t - 0.5 * g * t^2
 */
public class VerticalBallThrowWhileLoop2 {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    double x0 = 1.8d; // initial vertical position
    double v0 = 10d; // initial velocity upwards
    double g = 9.80665d; // free fall acceleration downwards

    int i = -1;
    while (++i < 12) { // the condition is checked at the top of the loop
      double t = 0.2d * i;
      System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints the current position
    }
  }
}
```

While-Loop with stdin-Combo

Listing: Vertical Ball Throw with reading t from stdin

```
import java.util.Scanner;

/**
 * A ball is thrown vertically upwards into the air by a x0m tall person<br/>
 * with velocity v0m/s. Where is it after t seconds?<br/>
 * x(t) = x0 + v0 * t - 0.5 * g * t^2 <br/>
 * Allow user to enter arbitrarily many values for t
 */
public class VerticalBallThrowWhileLoopSystemIn {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    double x0 = 1.8d; // initial vertical position
    double v0 = 10d; // initial velocity upwards
    double g = 9.80665d; // free fall acceleration downwards

    Scanner scanner = new Scanner(System.in); // initiate reading from System.in, ignore for now

    System.err.println("Keep entering times t, hit ctrl-d to exit."); //$NON-NLS-1$
    while (scanner.hasNext()) { // until stdin was closed with ctrl-d (or end has been reached)
      double t = scanner.nextDouble(); // read next number from stdin
      System.out.print("position at x("); //$NON-NLS-1$
      System.out.print(t); // print t
      System.out.print(")="); //$NON-NLS-1$
      double xt = x0 + (v0 * t) - 0.5d * g * t * t; // x(t) = x0 + v0 * t - 0.5 * g * t^2
      System.out.println((xt > 0d) ? xt : 0d); // prints result and makes sure the ball stops at ground
    }
  }
}
```

Do-While Loop

- `do { ...} while (condition)` loops are similar to `while` loops

Do-While Loop

- `do { ...} while (condition)` loops are similar to `while` loops
- The `condition` is checked *after* every iteration and another iteration is only performed if `condition` evaluates to true

Do-While Loop

- `do { ...} while (condition)` loops are similar to `while` loops
- The `condition` is checked *after* every iteration and another iteration is only performed if `condition` evaluates to true
- do-while loops always perform at least one iteration

Do-While-Loop Example

Listing: Vertical Ball Throw: $t = 0, 0.2, 0.4, \ldots, 2.2$, and $2.2s$.

```
/**
 * A ball is thrown vertically upwards into the air by a 1.8m tall person<br/>
 * with velocity 10m/s. Where is it after t=0,0.2,...,2.2 seconds?<br/>
 * x(t) = x0 + v0 * t - 0.5 * g * t^2
 */
public class VerticalBallThrowDoWhileLoop {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    double x0 = 1.8d; // initial vertical position
    double v0 = 10d; // initial velocity upwards
    double g = 9.80665d; // free fall acceleration downwards

    int i = 0;
    do { // using an integer for counting
      double t = 0.2d * i; // and multiplying it by 0.2 is more accurate
      System.out.println(x0 + (v0 * t) - 0.5d * g * t * t); // prints the current position
      i++;
    } while (i < 12); // the condition is checked at the bottom of the loop
  }
}
```

Summary

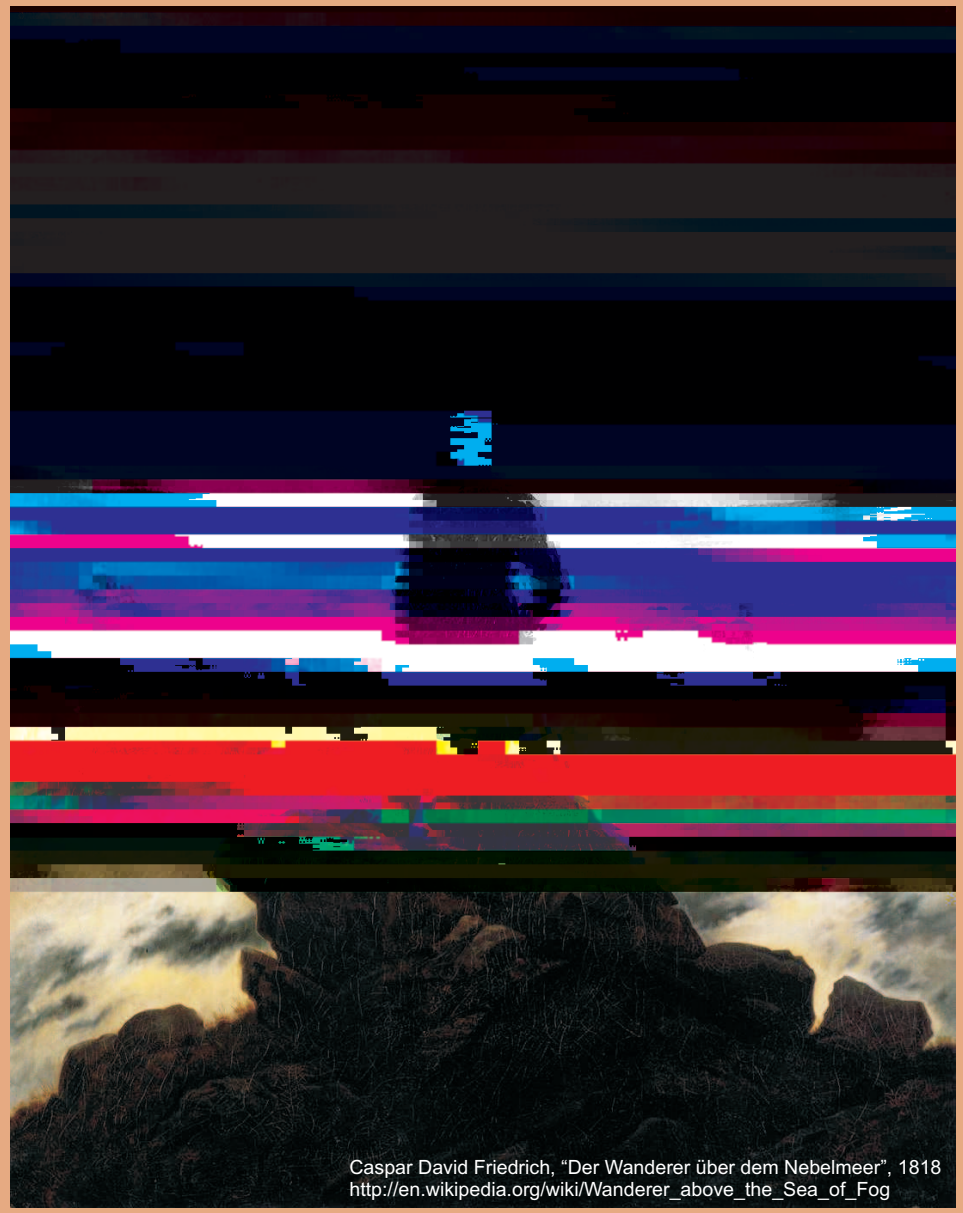
- We have learned how to repeat actions in a program.
- We have learned three types of loops for this purpose: for, while, and do-while loops.
- We have learned the special key words `break` and `continue` which can also be used to control loops
- We now know the most basic control flow primitives to structure programs which process input data

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China



Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog