合肥學院
HEFEI UNIVERSITY

OOP with Java
12. Static Variables

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

(Static) Variables

- We have seen how static methods can be defined

(Static) Variables

- We have seen how static methods can be defined
- If I define a static method inside class A, it is available to all the code in class A.

(Static) Variables

- We have seen how static methods can be defined
- If I define a static method inside class A, it is available to all the code in class A.
- (under some conditions) it is also available to the code in other classes

(Static) Variables

- We have seen how static methods can be defined
- If I define a static method inside class A, it is available to all the code in class A.
- (under some conditions) it is also available to the code in other classes
- Static variables follow the same concept: They are globally declared variables available to all the code in our class

(Static) Variables

- We have seen how static methods can be defined
- If I define a static method inside class A, it is available to all the code in class A.
- (under some conditions) it is also available to the code in other classes
- Static variables follow the same concept: They are globally declared variables available to all the code in our class
- (under some conditions) they become also available to the code in other classes

Declaring Global Variables

- A static/global variable can be declared anywhere in the class, but outside of the method body, i.e., in the class body itself

Declaring Global Variables

- A static/global variable can be declared anywhere in the class, but outside of the method body, i.e., in the class body itself
- The declaration is the same as for local variables, but `static` is prepended

Declaring Global Variables

- A static/global variable can be declared anywhere in the class, but outside of the method body, i.e., in the class body itself
- The declaration is the same as for local variables, but `static` is prepended
- Static variables can be initialized, modified, set exactly like local variables

Declaring Global Variables

- A static/global variable can be declared anywhere in the class, but outside of the method body, i.e., in the class body itself
- The declaration is the same as for local variables, but `static` is prepended
- Static variables can be initialized, modified, set exactly like local variables
- If we also mark them with the keyword `final` and initialize them, they can never be changed, i.e., are constants. Constants usually have all-uppercase names

Declaring Global Variables

- A static/global variable can be declared anywhere in the class, but outside of the method body, i.e., in the class body itself
- The declaration is the same as for local variables, but `static` is prepended
- Static variables can be initialized, modified, set exactly like local variables
- If we also mark them with the keyword `final` and initialize them, they can never be changed, i.e., are constants. Constants usually have all-uppercase names
- We can access static variables in other classes by using "`canonical-name-of-class.name-of-variable`"

Vertical Ball Throw with Constant for g

Listing: Vertical Ball Throw with Constant for g

```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 VerticalBallThrowFunctionAndConstants {

  /** make the constant G available to our code. "final" means that G can never be changed.*/
  static final double G = 9.80665d;

  /** Compute the position of a ball (good style: these comments document
   *                                                what the method does)
   * @param x0 the height of the thrower, i.e., the initial vertical position
   * @param v0 the vertical upward velocity with which the ball is thrown
   * @param t the time at which we want to get the position x(t)
   * @return the position x(t) of the ball at time step t
   */
  static double position(double x0, double v0, double t) {
    return x0 + (v0 * t) - 0.5d * G * t * t;
  }

  /** The main routine
   * @param args
   *          we ignore this parameter */
  public static final void main(String[] args) {
    for (int i = 0; i < 12; i++) { // using an integer for counting
      System.out.println(position(1.8d, 10d, 0.2d * i)); // prints the current position
    }
  }
}
```

Listing: Faster Fibonacci Numbers with Cache

```
/** An example program computing Fibonacci numbers F(n) = F(n-1) + F(n-2)
 * with F(1) = F(2) = 1 recursively using a cache for faster computation. */
public class FibonacciRecursiveCached {

  /** a cache variable */
  static long[] CACHE = new long[1000];

  /**
   * Recursively compute the ith Fibonacci number
   *
   * @param i
   *          the index of the number to compute
   * @return ith Fibonacci number
   */
  static long F(int i) {
    if ((i == 1L) || (i == 2L)) {
      return 1; // take care of cases F(1) and F(2)
    }
    if (i < CACHE.length) { // is i small enough to use the cache?
      if (CACHE[i] > 0) { // has F(i) already been computed ?
        return CACHE[i]; // yes, then we can directly return it
      }
      return CACHE[i] = F(i-1) + F(i-2); // no? recurse and cache result
    }
    return F(i-1) + F(i-2); // i is too big, just recurse
  }

  /** The main routine
   * @param args
   *          we ignore this parameter */
  public static final void main(String[] args) {
    for (int i = 1; i <= 90; i++) { // print the first 90 Fibonacci numbers
      System.out.print("F("); //$NON-NLS-1$
      System.out.print(i);
      System.out.print(")␣=␣"); //$NON-NLS-1$
      System.out.println(F(i));
    }
  }
}
```

Using the Constants from Java-Provided Class Math

Listing: Using the constants for π and e

```
/** An example program using the methods of java.lang.Math */
public class MathMethodsAndConstants {

  /** The main routine
   * @param args
   *             we ignore this parameter */
  public static final void main(String[] args) {
    System.out.println(Math.log(Math.E));
    System.out.println(Math.sin(Math.PI / 2d));

    // ok, the one below is not from Math but from our class
    // VerticalBallThrowFunctionAndConstants
    System.out.println(VerticalBallThrowFunctionAndConstants.G);
  }
}
```

Summary

- We have learned what static variables are.
- We have learned how to define them and how to use them.
- We have learned that we can put static variables into different classes and use static variables from different classes.

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