13 Methods

Instructor: Ke Wei（柯韋）

➽ A319 ✆ Ext. 6452 ✉ wke@ipm.edu.mo

http://brouwer.ipm.edu.mo/COMP112/18/

Bachelor of Science in Computing, School of Public Administration, Macao Polytechnic Institute

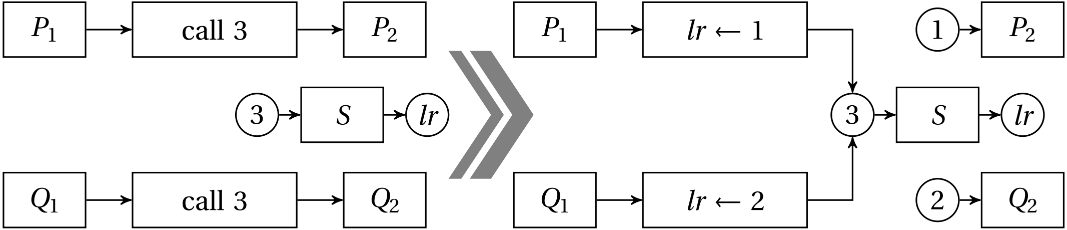
October 19, 2018

Outline

1. Subroutines, Procedures and Functions
2. Methods
3. Static Methods
4. Local Variables
5. Reading Homework

Subroutines

- A particular task could be performed a number of times on different data values.
- This repeated task is normally implemented as a subroutine.
- A new branching instruction to support subroutines: branching to a location stored in a *variable*, such as a register *lr*.
- **Call**: before we go to the subroutine, we store the location of the next instruction to *lr*.
- **Return**: when the subroutine finishes, we fetch the location from *lr* and go back.



- How about nested subroutine calls?

Procedures, Functions and Methods

- In Java, methods act as subroutines, procedures and functions.
- Subroutines are common groups of statements that are shared in different places.

```
void printBar() { System.out.println("--------"); }
```

- Procedures are subroutines that may formally accept arguments, such that a procedure may be called upon different values.

```
void printBar(int n) {
    for ( int i = 0; i < n; ++i ) System.out.print("-");
    System.out.println();
}
```

- Functions are procedures that may return values.

```
int max(int x, int y) { return x > y ? x : y; }
```

- In mathematics, functions do not have side effects, they just map arguments to results.
- In Java, methods usually have side effects, that they can change variables outside and perform I/O operations.

Defining a Method

- Every method in Java belongs to a class. We must define a method within a class, such as *MyClass*.
- Every method has a signature (header), which mentions the method name, the types of the parameters and the return value.

```
int sumSqr(int x, int y)
```

- A statement block following the method header defines the method body.

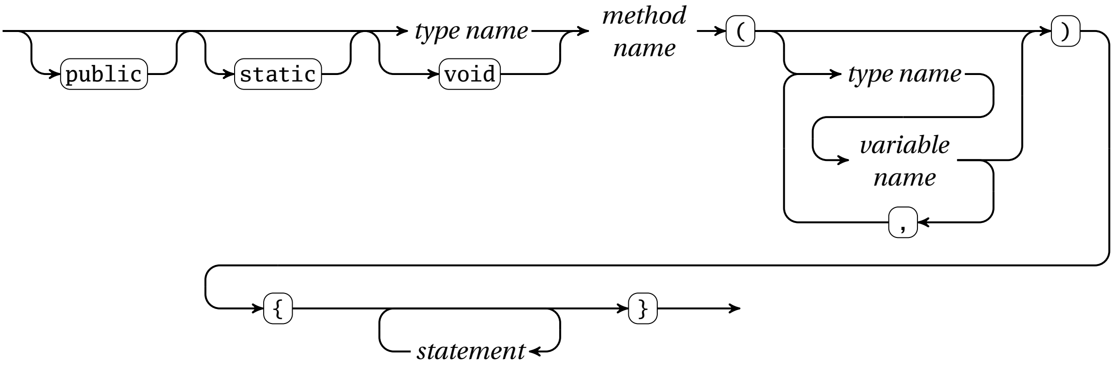
```
{ return sqr(x) * sqr(y); }
```

- The task that the method performs is specified in the block.
- A method returns when the `return` statement is executed, or when the execution reaches the end of the block if the method has no return value, that is, the return type is `void`.

```
void doubleStarBars() {
    for ( int i = 0; i < 80; ++i )
        System.out.print((i+1)%40 == 0 ? "*\n" : "*");
}
```

Syntax Diagram of Method Definitions

method definition



Calling a Method

- A method call starts with an object, say *myObj*, of the class that defines the method, followed by the method name and arguments.

```
int a = 100 + myObj.sumSqr(100, 200);
```

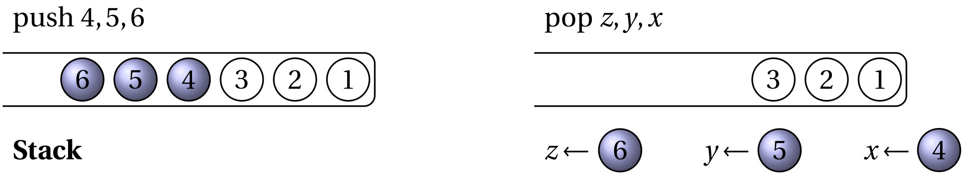
- A method call is an expression, if it returns a value. The value returned is the result of this expression. You can put it anywhere that an expression fits.
- A method with the `void` return type must be called as a statement.

```
myObj.doubleStarBars();
```

- The arguments (100 and 200) are assigned to the parameters (x and y) declared in the signature, they can be used as variables in the method.
- The object (*myObj*) is assigned to an implicit parameter called "`this`", it can be used as a constant in the method to refer to the calling object.
- The result is stored in a special variable, for example, a register r_0, that can be fetched when the method returns.

What Really Happen?

- The registers to store return addresses and return values are *lr* and r_0 in all method calls.
- What if we call another method while we are in a method, just like calling *sqr* in *sumSqr*?
- What if we call a method having x as a parameter from a method that declares x as a variable?
- All these issues can be solved by saving the duplicate variables to a stack when entering a subroutine,
- And, restoring the duplicate variables from the stack when leaving the subroutine.



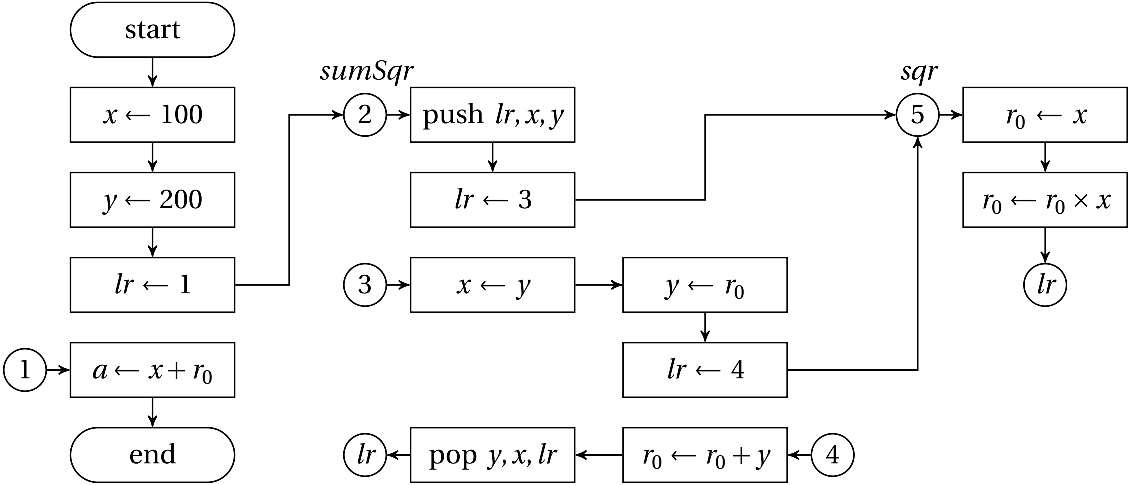
push $4, 5, 6$

Stack

pop z, y, x

$z \leftarrow 6$ $y \leftarrow 5$ $x \leftarrow 4$

Preserving Variables for Nested Method Calls



start

$x \leftarrow 100$

$y \leftarrow 200$

$lr \leftarrow 1$

1 $a \leftarrow x + r_0$

end

sumSqr

2 push lr, x, y

$lr \leftarrow 3$

3 $x \leftarrow y$

$y \leftarrow r_0$

$lr \leftarrow 4$

4 $r_0 \leftarrow r_0 + y$

pop y, x, lr

lr

sqr

5 $r_0 \leftarrow x$

$r_0 \leftarrow r_0 \times x$

lr

Static Methods

- Usually, a method must be called upon an object instance of the class defining the method. Such a method is called an *instance* method. For examples,

```
int i = scanner.nextInt();
int ch = r.read();
```

- The object instance is passed to the method via an implicit parameter "`this`" of the type of the class.
- If a method does not need the object instance (`this`), it can be declared `static`.

```
public static int multiply(int x, int y) { return x * y; }
```

- A static method can be called with the class name `MyClass.multiply(10,20)` rather than the object instance of the class.
- If you call a method within the defining class, you can omit the prefix. For instance methods, "`this`" is the default prefix, for static methods, the class name is the default prefix.

Local Variables

- Variables declared in a block is local to the block and its inner blocks.
- Outer blocks cannot see variables declared in inner blocks.

```
{ int o = 2; { int i = o;✔... } { int x = i;✘... } int y = i;✘... }
```

- Two parallel blocks cannot see variables declared in the other block.
- Variables declared inside the method body are local to the method.
- A name can be used in different blocks for different variables at the same time, inner names hide outer names.
- Where a variable is visible is called its *scope*.
- Each time the execution enters a block, a new set of local variables of the block is created, they are destroyed when the execution leaves the block.
- Method parameters are local variables with the widest scope in the method.

Reading Homework

Textbook

- Section 6.1–6.5, 6.9.

Internet

- Subroutine (`http://en.wikipedia.org/wiki/Subroutine`).
- Local variable (`http://en.wikipedia.org/wiki/Local_variable`).

Self-test

- 5.1 – 5.14, 5.19 – 5.20 (`http://tiger.armstrong.edu/selftest/selftest9e?chapter=5`).