Jasmin Instruction Part 3

Csc3210
Yuan Long

Summary

• Subroutine
• Method
What is subroutine?

- A subprogram (a block of instruction) in \textbf{current method}.
- Performs a fixed task.
- \textbf{Share} same operand stack and local variable array with current method.
- Codes can be copied to the called location without modification.

What is method?

- A subprogram \textbf{belonging to class}.
- Each method has its \textbf{own operand stack and local variable array}.
- Invoked by class name (for static method) or instance.
Subroutine

• Example 1: Print variable a, b and c’s value

Pseudocode:

PrintValueA:
   iload_1
   ; Print out value of a
PrintValueB:
   iload_2
   ; Print out value of b
PrintValueC:
   iload_3
   ; Print out value of c

How to print out a variable’s value(integer)?
Assume value is in local variable 1.

getstatic java/lang/System/out Ljava/io/PrintStream;
iloa_1
invokevirtual java/io/PrintStream/println(I)V
Subroutine

• Example 1: Print variable a, b and c’s value

Then we can fill out the previous pseudocode like below:

PrintValueA:
    getstatic java/lang/System/out Ljava/io/PrintStream;
    iload_1
    invokevirtual java/io/PrintStream/println(I)V

PrintValueB:
    getstatic java/lang/System/out Ljava/io/PrintStream;
    iload_2
    invokevirtual java/io/PrintStream/println(I)V

PrintValueC:
    getstatic java/lang/System/out Ljava/io/PrintStream;
    iload_3
    invokevirtual java/io/PrintStream/println(I)V

Subroutine

• Example 1: Print variable a, b and c’s value
  – However, if we need to print out another 3 or more variable’s value, getstatic and invokevirtual should be repeated more times.
  – Then try to define a block of program and call that block.
  – We name that block as PrintString.
Subroutine

• Example 1: Print variable a, b and c’s value

A new program could be like below.

PrintValueA:
  iload_1
  goto PrintString
PrintValueB:
  iload_2
  goto PrintString
PrintValueC:
  iload_3
  goto PrintString
PrintString:
  ; take values from stack
  ; and perform some operations
  ; finished....
  goto ....??

Subroutine

• Example 1: Print variable a, b and c’s value

– The above program requires three basic modifications:

  1) Record the original location (the value of PC before the jump).
     astore_N ; store the original location in local
                 ; variable N

  2) Return to the original location.
     ret      N ; return to the location stored in local
                 ; variable N

  3) A special instruction to call the block (subroutine)
     jsr      SubroutineName
Subroutine

• How to define a subroutine?

At a certain location of your program, use the following form to add a subroutine.

```java
SuboutineName:
    astore_N ; store return address (original location)
    ; in local variable N.
    ; do something
    ret N ; return to the location stored in local
            ; variable N.

; other instructions in method
... .end method
```

Subroutine

• Example 1: Print variable a, b and c’s value

Final code:

```java
PrintValueA:
    iload_1
    jsr PrintString
PrintValueB:
    iload_2
    jsr PrintString
PrintValueC:
    iload_3
    jsr PrintString
PrintString:
    astore_4
    getstatic java/lang/System/out Ljava/io/PrintStream;
    swap ; put the arguments in the right order
    invokevirtual java/io/PrintStream/println(I)V
    ret 4
```
Subroutine

• Example 2: Random Number Generator

[From Patrick 2007]

– Generate random float number over the range of 0 to 1

\[ \text{newvalue} = (a \cdot \text{oldvalue} + c) \mod m \]

After each generation, \( \text{oldvalue} \) is replaced by \( \text{newvalue} \).

\[ \text{floatNumber} = \frac{\text{newvalue}}{m} \]

```java
; compute a*oldvalue:
ldc2_w 65537    ; a = 2^16+1
iload_3
i2l
imul

; add c         ; c = 5
ldc2_w 5
ladd

; and take the remainder mod m
ldc2_w 2147483647 ; m = 2147483647
lrem
i2l
dup
istore_3

i2f
ldc 2147483647.0
fdiv
```
Example 3: Monte Carlo Estimation of \( \pi \)

[From Patrick 2007]

- \( \pi = 3.14159 \ldots \)
- How to figure out the value for mathematicians?
  - Many approaches, but a notable and simple one is called Monte Carlo Estimation.

**General Idea of Monte Carlo Estimation of \( \pi \)**

- **Throwing darts** (randomly and uniformly) at a square shape board (side = 2).
- A circle \((r=1)\) is drawn within the square.
- The darts can also land inside the circle.
- **Possibility of “hit” in circle** = Area of circle / Area of Square = \( \pi /4 \)

In other words, if we throw 100000 darts, we would expect 100000 * (\( \pi /4 \)) of them to land inside of circles.
Subroutine

• Algorithm
  Hit_count = 0
  For ( darts_count = 1 : 100000)
    Randomly generate \((x,y)\) position for new darts
    If \((x,y)\) inside circle
      Hit_count ++;
    End if
  End for
  Possibility = Hit_count / 100000
  \(\Pi = \text{Possibility} \times 4\)

Subroutine

• Implementation
  – Problem 1:
    How to randomly generate \((x,y)\)?
    We know that \(-1<=x<=1\, , \, -1<=y<=1\)
    Randomly generate a number from a range.
  – Problem 2:
    How to check \((x,y)\) inside circle?
    \(x^2 + y^2 <=1\)
Subroutine

• Code for problem 1

We use a subroutine for it.

Random:

; compute a\text{*oldvalue}:
ldc2_w 65537 ; a = 2^{16}+1
iload_3
i2l
imul

; add c ; c = 5
ldc2_w 5
ladd

; and take the remainder mod m
ldc2_w 2147483647 ; m = 2147483647
lrem
i2i
dup
istore_3

i2f
ldc 2147483647.0
fdiv
ret 6

Subroutine

• Code for problem 2

If (x^2 + y^2 <= 1) hit\_count++;

fload_4
dup
fmul
fload_5
dup
fmul
fadd
fconst_1
fcmp
iflt endif
iinc 2 1
endif:
Subroutine

• Final Code
  example_3.j

Methods

• Static methods
  invokestatic

• Non-static methods
  – Constructor
    invokespecial
  – Other
    invokevirtual, invokespecial
Method

• General syntax to invoke a method

```
Invoke?  Method_info
```

```
invokevirtual  java/io/PrintStream/println(I)V
```

Method

• Descriptor

  – Describe the types of variables for fields in class, parameters and return values in a method

<table>
<thead>
<tr>
<th>BaseType Character</th>
<th>Type</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>byte</td>
<td>signed byte</td>
</tr>
<tr>
<td>C</td>
<td>char</td>
<td>Unicode character code point in the Basic Multilingual Plane, encoded with UTF-16</td>
</tr>
<tr>
<td>D</td>
<td>double</td>
<td>double-precision floating-point value</td>
</tr>
<tr>
<td>F</td>
<td>float</td>
<td>single-precision floating-point value</td>
</tr>
<tr>
<td>I</td>
<td>int</td>
<td>integer</td>
</tr>
<tr>
<td>J</td>
<td>long</td>
<td>long integer</td>
</tr>
<tr>
<td>L ClassName ;</td>
<td>reference</td>
<td>an instance of class ClassName</td>
</tr>
<tr>
<td>S</td>
<td>short</td>
<td>signed short</td>
</tr>
<tr>
<td>Z</td>
<td>boolean</td>
<td>true or false</td>
</tr>
<tr>
<td>[</td>
<td>reference</td>
<td>one array dimension</td>
</tr>
</tbody>
</table>
Method

• Descriptor

```
.method public static main([Ljava/lang/String;)V
```

Methods

• How the stack works?
  – Once invoke family instructions is executed,
    • Check the descriptor
    • Decide how many arguments the method takes
    • Pop arguments off stack
    • Pop objectref off stack. (*Not required for static method*)

<table>
<thead>
<tr>
<th>Argument N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argument N-1</td>
</tr>
<tr>
<td>.....</td>
</tr>
<tr>
<td>Argument 1</td>
</tr>
<tr>
<td><em>ObjectRef</em></td>
</tr>
</tbody>
</table>
Method

• Example4.java

```java
public class Example4{
    public int a;
    public int b;
    public int c;
    public Example4(){
        a = 0;
        b = 0;
        c = 0;
    }
    public Example4(int x, int y, int z){
        a = x;
        b = y;
        c = z;
    }
    public static void PrintValue(int x){
        System.out.println(x);
    }
    private void setA(int new_value){
        a = new_value;
    }
    public int getA(){
        return a;
    }
    public int max(int x, int y){
        if (x>y) return x;
        else return y;
    }
}
```

Method

• Example4.java

```java
public static void main(String args[]){
    Example4.PrintValue(100); // call the static method
    Example4 e1 = new Example4(1,2,3); // call the constructor

    int valueA=0;
    valueA=e1.getA(); // call a non-static and non-constructor method
    System.out.println("Original value for a = ");
    System.out.println(valueA);

    e1.setA(5);
    valueA=e1.a; // access a field in class
    System.out.println("After setA(), the new value for a= ");
    System.out.println(valueA);
}
```
Static Methods

• Define a static method

```java
; public static void PrintValue(int)

.method public static PrintValue(I)V
 .limit stack 3
 .limit locals 2
 ; the integer is in local variable 0
 ; ...
 return
 .end method
```

Static Methods

• Call a static method

  – Call PrintValue with a parameter of 100. This goes in the main method.

```
Example4.PrintValue(100);
// call the static method
```

```
bipush 100
invokestatic Example4/PrintValue(I)V

;Example4 is the class name.
```
Non-static Methods

• Define a constructor

; public void example4(int x, int y, int z)

.method public <init>(III)V
.limit stack 3
.limit locals 5
; ...
return
.end method

Non-static Methods

• Call a constructor

– Call constructor example4() with parameters 1, 2, 3.

– To call it in the main method, we should new one object first.

Example4 e1 = new Example4(1, 2, 3);
// call the constructor

new example4(); ; new one object and put object reference
; on top of stack

iconst_1
iconst_2
iconst_3
invokepecial example4/<init>(iii)V
atore_1 ; put the reference of instance into local variable 1
Non-static methods

• Define other methods

    ; public void getA()

    .method public getA()I
     .limit stack 3
     .limit locals 2
     ; the integer is in local variable 0
     ; ...
     return
    .end method

Non-static methods

• Call other methods

  – Call getA()
  – Assume the reference of an object of Example4 is stored in local variable 1

     aload_1
    invokevirtual Example4/getA()I

     ; Example4 is the class name.
Fields

• Define a filed

; public int a
.field public a I

Fields

• Retrieve value from a field
  – For static filed
    getstatic field_info descriptor
  – For other
    getfield field_info descriptor

Field_info : [class_name]/[Field_name]
Descriptor : Type of field
Fields

- Set value for a field
  - For static field
    `putstatic field_info descriptor`
  - For other
    `putfield field_info descriptor`

Field_info : [class_name]/[Field_name]
Descriptor : Type of field

```
; valueA=e1.a
getfield Example4/a I;

; e1.a =3
putfield Example4/a I;
```

;PrintStream obj = java.lang.System.out;
getstatic java/lang/System/out Ljava/io/PrintStream;

References