CS 153: Concepts of Compiler Design

November 16 Class Meeting

Department of Computer Science
San Jose State University

Fall 2017
Instructor: Ron Mak
www.cs.sjsu.edu/~mak

A Free Compiler Book!

- A more "traditional" compiler textbook first published in 1979:

  http://www.eis.mdx.ac.uk/staffpages/r_bornat/books/compiling.pdf

- A bit dated, but free to download.
  - 2.6 MB PDF.

The Pascal Runtime Library

- Useful utility classes written in Java that contain methods your compiled Jasmin code can call.
  - Create an archive file `PascalRTL.jar`.
  - See Chapters 16 and 17.
- You don’t have to know Java to use this library. Just follow the examples of how to instantiate the classes and call their methods (member functions).

The Pascal Runtime Library Utility Classes

- `PascalTextIn`
  - Runtime text input based on the scanner.
- `RunTimer`
  - Time the execution of compiled programs and print the elapsed run time.
- `RangeChecker`
  - Perform runtime range checking.
- `PascalRuntimeException`
  - Error exception thrown while executing compiled code.
- `PaddedString`
  - Pascal string implementation with blank-padding

The Pascal Runtime Library, *cont'd*

- The Pascal Runtime Library can reuse some classes from the front end.
- When a Pascal program calls the standard procedure **`read`** to input values at run time, **`read`** must scan the input text for values of various data types (integer, real, string, etc.).
- Therefore, reuse the scanner and token classes by including them in the library.

The Pascal Runtime Library, *cont'd*

- Include the runtime library on your class path with **`-cp`** when you run your compiled program.
  - Mac and Linux:

```
java –cp .:PascalRTL.jar MyProgram
```

  - Windows:

```
java –cp .;PascalRTL.jar MyProgram
```

- Your generated code can call routines in the library.

The Pascal Runtime Library, *cont’d*

- `Cloner.class`
- `PaddedString.class`
- `PascalRuntimeException.class`
- `PascalTextIn.class`
- `RangeChecker.class`
- `RunTimer.class`
- `BWrap.class`
- `CWrap.class`
- `IWrap.class`
- `RWrap.class`

The Pascal Runtime Library, *cont'd*

- `wci/frontend/EofToken.class`
- `wci/frontend/Scanner.class`
- `wci/frontend/Source.class`
- `wci/frontend/Token.class`
- `wci/frontend/TokenType.class`
- `wci/frontend/pascal/PascalScanner.class`
- `wci/frontend/pascal/PascalToken.class`
- `wci/frontend/pascal/PascalTokenType.class`
- `wci/frontend/pascal/tokens/PascalErrorToken.class`
- `wci/frontend/pascal/tokens/PascalNumberToken.class`
- `wci/frontend/pascal/tokens/PascalSpecialSymbolToken.class`
- `wci/frontend/pascal/tokens/PascalStringToken.class`
- `wci/frontend/pascal/tokens/PascalWordToken.class`
- `wci/message/Message.class`
- `wci/message/MessageHandler.class`
- `wci/message/MessageListener.class`
- `wci/message/MessageProducer.class`
- `wci/message/MessageType.class`

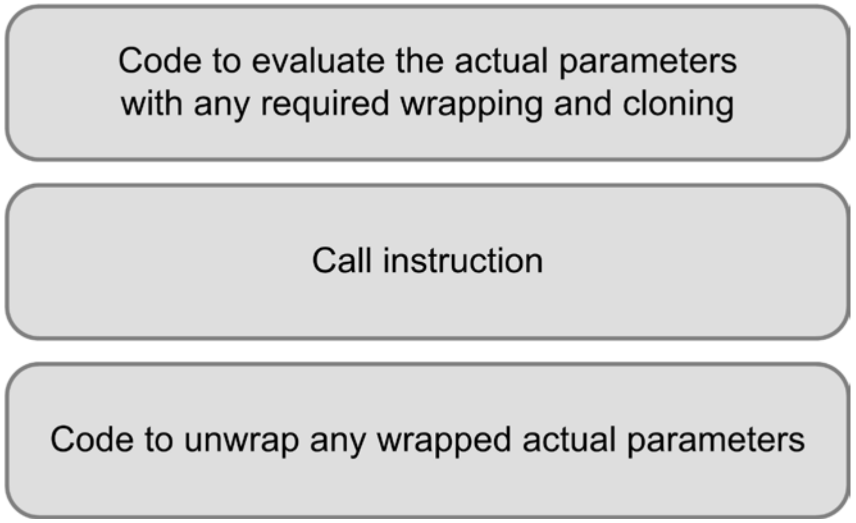
Pascal Parameter Passing

- Pass by value
- Pass by reference (“VAR parameters”)

True or False?

- Java passes <u>scalar parameters</u> by <u>value</u>.

Passing Parameters

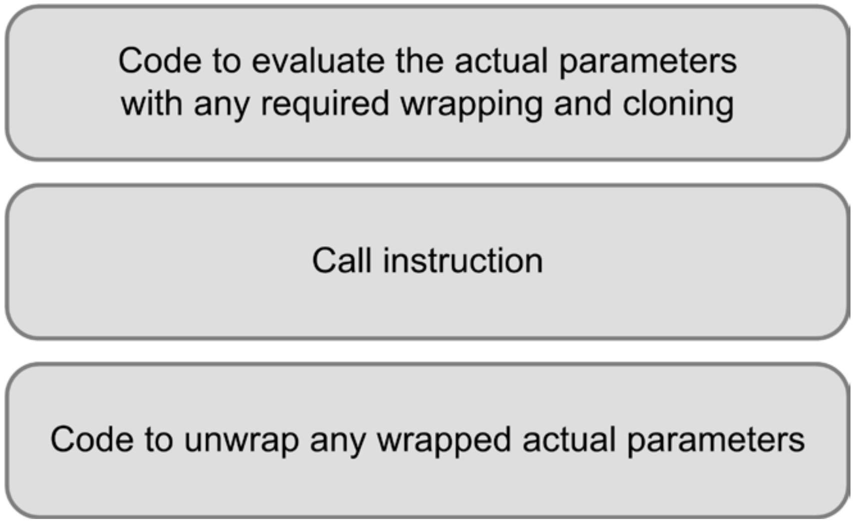

Code to evaluate the actual parameters with any required wrapping and cloning
Call instruction
Code to unwrap any wrapped actual parameters

- Pascal can pass parameters by value and by reference.
  - VAR parameters = pass by reference
- Java and Jasmin pass scalar values by value.
- To pass a scalar value by reference, you must first wrap the value inside an object.
  - Pass the object reference (by value) to the routine.
  - The routine can modify the wrapped value.
  - Upon return, the caller must unwrap the changed value.

True or False?

- Java passes object parameters by reference.

Passing Parameters, *cont'd*


Code to evaluate the actual parameters with any required wrapping and cloning
Call instruction
Code to unwrap any wrapped actual parameters

When a formal parameter to a method is a reference to an object, the method can change the value of the object (such as by modifying the values of the object fields). But the method cannot change the parameter's value to refer to another object and have the method's caller see that change.

- Java and Jasmin pass references to objects by value.
- To pass an array or record by value as in Pascal, first clone the array or record value and then pass the reference to the clone.

Passing Parameters, *cont'd*

- The Pascal Runtime Library contains classes for passing parameters by value or by reference.

- Classes **`BWrap`**, **`CWrap`**, **`IWrap`**, and **`RWrap`** wrap a boolean, character, integer, and real scalar value, respectively, to be passed by reference.

- Class **`Cloner`** clones an array or record to be passed by value.

```
public class IWrap
{
    public int value;

    public IWrap(int value)
    {
        this.value = value;
    }
}
```

See WCI Chapter 17 for the details.

Example: Passing Scalars by Reference

```
PROGRAM parmswrap;

VAR
  i, j : integer;

PROCEDURE swap(VAR parm1, parm2
                   : integer);
  VAR
    temp : integer;

  BEGIN
    temp  := parm1;
    parm1 := parm2;
    parm2 := temp;
  END;

BEGIN
  i := 10;
  j := 20;
  swap(i, j);
  writeln('Result: i = ', i:0,
          ', j = ', j:0);
END.
```

```
.method public static main([Ljava/lang/String;)V
  ...
  new    IWrap
  dup
  getstatic         parmswrap/i I
  invokenonvirtual IWrap/<init>(I)V
  dup
  astore_1
  new    IWrap
  dup
  getstatic         parmswrap/j I
  invokenonvirtual IWrap/<init>(I)V
  dup
  astore_2

  invokestatic   parmswrap/swap(LIWrap;LIWrap;)V

  aload_1
  getfield       IWrap/value I
  putstatic      parmswrap/i I
  aload_2
  getfield       IWrap/value I
  putstatic      parmswrap/j I
```

Wrap **i**.

Allocate slots #1 and #2 as temporaries to store the wrapped **i** and **j**.

Wrap **j**.

Call method.

Unwrap **i**.

Unwrap **j**.

Example: Passing Scalars by Reference, *cont'd*

```
PROGRAM parmswrap;

VAR
  i, j : integer;
                          #0       #1
PROCEDURE swap(VAR parm1, parm2
                   : integer);
  VAR
 #2temp : integer;

  BEGIN
    temp  := parm1;
    parm1 := parm2;
    parm2 := temp;
  END;

BEGIN
  i := 10;
  j := 20;
  swap(i, j);
  writeln('Result: i = ', i:0,
          ', j = ', j:0);
END.
```

```
.method private static swap(LIWrap;LIWrap;)V

.var 2 is temp  I
.var 0 is parm1 LIWrap;
.var 1 is parm2 LIWrap;

    aload_0
    getfield  IWrap/value I
    istore_2

    aload_0
    aload_1
    getfield  IWrap/value I
    putfield  IWrap/value I

    aload_1
    iload_2
    putfield  IWrap/value I

    return

.limit locals 3
.limit stack 2
.end method
```

Access the wrapped values of **parm1** and **parm2** and swap them.

Example: Passing an Array by Value

```
PROGRAM parmsclone;

TYPE
  cube = ARRAY [0..1, 0..2, 0..3] OF integer;

VAR
  vvv : cube;

PROCEDURE printCube(VAR c : cube);
  ...

PROCEDURE doCubeValue(c : cube);
  VAR
    i, j, k : integer;

  BEGIN
    FOR i := 0 TO 1 DO BEGIN
      FOR j := 0 TO 2 DO BEGIN
        FOR k := 0 TO 3 DO BEGIN
          c[i,j][k] := 200*i + 10*j +k;
        END;
      END;
    END;

    writeln('In doCubeValue:');
    printCube(c);
  END;
```

```
PROCEDURE doCubeRef(VAR c : cube);
  ...
  BEGIN
    ...
      c[i,j][k] := 100*i + 10*j +k;
    ...
  END;

BEGIN
  doCubeRef(vvv);

  writeln('In main:');
  printCube(vvv);

  doCubeValue(vvv);

  writeln('In main:');
  printCube(vvv);
END.
```

```
getstatic      parmsclone/vvv [[[I
invokestatic   Cloner.deepClone(Ljava/lang/Object;)
                               Ljava/lang/Object;
checkcast      [[[I
invokestatic   parmsclone/docubevalue([[[I)V
```

Class Cloner

- In the Pascal Runtime Library:

```
public class Cloner
{
  public static Object deepClone(Object original)
    throws PascalRuntimeException
  {
    try {
      ByteArrayOutputStream baos = new ByteArrayOutputStream();
      ObjectOutputStream oos = new ObjectOutputStream(baos);
      oos.writeObject(original);

      ByteArrayInputStream bais =
          new ByteArrayInputStream(baos.toByteArray());
      ObjectInputStream ois = new ObjectInputStream(bais);

      return ois.readObject();
    }
    catch (Exception ex) {
      throw new PascalRuntimeException("Deep clone failed.");
    }
  }
}
```

Write the original object to a byte array stream.

Construct a copy from the stream.

Return the copy as the deep clone, too.

Wrapping is not a Perfect Solution

- Wrapping a scalar value as an object to be passed by reference is not a perfect solution.
- Wrapping is actually closer to "passing by value-result".

Demo

Compilation Demos

- Wolf Island
- Hilbert Matrix
  - Compare execution speeds: interpreter vs. compiler
  - How much faster is the compiled code?