Pascal To Java: The Transition

Documentation

Variables and constants

Advanced input and output

Decision making

Loops

Homogeneous composite types (arrays)

Modularization (functions/procedures and methods)

Basic Structure Of A Java Program

```
class <name of class>
{
   public static void main (String[] args)
   {
   }
}
```

Documentation / Comments

Pascal

(* Start of documentation

*) End of documentation

Java

Multi-line documentation

/* Start of documentation

*/ End of documentation

Documentation for a single line

// Everything until the end of the line is treated as a comment

Variable Declaration

```
class <name of class>
{
   public static void main (String[] args)
   {
         // Variable declarations occur here
   }
}
```

Some Simple Types Of Variables In Java

Type	Description
byte	8 bit signed integer
short	16 but signed integer
int	32 bit signed integer
long	64 bit signed integer
float	32 bit signed integer
double	64 bit signed integer
char	16 bit Unicode character
Boolean	1 bit true or false value
String	A sequence of characters between double quotes ("")

Java Vs. Pascal Variable Declarations

Pascal

Format:

<*variable name*> : variable type;

Example:

num : integer;

Java

Format:

variable type <*variable name*>;

Example:

long num1;

double num2 = 2.33;

Constants In Java

Format

```
final <variable type> <variable name> = <value>;
```

Example

```
final int SIZE = 100;
```

The Semicolon In Pascal

Pascal

- Used to separate statements within a block
- This is okay in Pascal:

```
program test (output);
begin
 writeln("one");
 writeln("two")
end.
```

The Semicolon In Java

Java
• Follows each statement
• This is not okay in Java:

```
class BadExample
{
    public static void main (String [] argv)
    {
        System.out.println("one");
        System.out.println("two")
    }
}
```

Java Variables And Constants: A Small Example

```
class SmallExample
{
   public static void main (String [] argv)
   {
      final double PI = 3.14;
      String str = "Hosta la vista baby!";
      long num = 10;
      double db;
      db = PI;
      System.out.print("num=" + num);
      System.out.println(" db=" + db);
      System.out.println("Str says..." + str);
   }
}
```

Output : Some Escape Sequences

Escape sequence	Character value
\t	Horizontal tab (5)
\n	New line
\”	Double quote
\\	Backslash

Advanced Output

Employ the predefined code in TIO

To use:

(In Unix):

- Copy all the files in the directory /home/profs/tamj/tio to the directory where your Java program resides

(In your Java program include the following statements):

- import tio.*;
- FormattedWriter out = new FormattedWriter(System.out);

Advanced Output (2)

Statement	Effect
out.printf(<*variable or string1* > + <*variable or string 2*> …); MUST EVENTUALLY BE FOLLOWED BY A PRINTFLN!	Prints contents of field
out.printfln((<*variable or string1* > + <*variable or string 2*> …);	Prints contents of field and a newline
out.setWidth(<*integer value*>);	Sets the width of a field
out.setDigits(<*integer value*>);	Sets the number of places of precision
out.setJustify(out.LEFT); outsetJustify(out.RIGHT);	Left or right justify field contents

Advanced Output: Example

```
import tio.*;
class OutputExample
{
   public static void main (String [] argv)
   {
      FormattedWriter out = new FormattedWriter(System.out);
      int num = 123;
      double db = 123.45;
      out.setJustify(out.LEFT);
      out.setWidth(6);
      out.setDigits(1);
      out.printf("Start line");
      out.printf(num);
      out.printf(db);
      out.printf("End of line");
      out.printfln("");
   }
}
```

Advanced Input

Employ the predefined code in TIO

To use:

(In Unix):

- Copy all the files in the directory /home/profs/tamj/tio to the directory where your Java program resides

(In your Java program include the following statement):

- import tio.*;

Introduction to CPSC 233 James Tam

Advanced Input: Getting The Input

Console.in.readInt()	Reads in an integer Returns an integer
Console.in.readLong()	Reads in a long Returns a long
Console.in.readFloat()	Reads in a float Returns a float
Console.in.readDouble()	Reads in a double Returns a double
Console.in.readLine()	Reads in a line Returns a String
Console.in.readWord()	Reads in a word Returns a String
Console.in.readChar()	Reads in a character Returns an integer

Introduction to CPSC 233 James Tam

Relational Operators

Operator	Meaning of the operator
<	Less than
<=	Less than, equal to
>	Greater than
>=	Greater than, equal to
==	Equal to
!=	Not equal to

Common Logical Operators

&&	Logical AND
||	Logical OR
!	Logical NOT

Decision Making

Pascal

- If-then
- If-then-else
- If-then, else-if
- Case-of

Java

- If
- If, else
- If, else-if
- Switch

Decision Making: If

Format:

```
  if (Boolean Expression)
      Body
```

Example:

```
  if (x != y)
      System.out.println(“X and Y are not equal”);

  if ((x > 0) && (y > 0))
  {
      System.out.println();
      System.out.println("X and Y are positive");
  }
```

Decision Making: If, Else

Format

```
if (Boolean expression)
    Body of if
else
    Body of else
```

Example

```
if (x < 0)
    System.out.println("X is negative");
else
    System.out.println("X is non-negative");
```

Introduction to CPSC 233 James Tam

If, Else-If

Format

```
if (Boolean expression
    Body of If
else if (Boolean expression)
    Body of first else-if
    :    :         :
else if (Boolean expression)
    Body of last else-if
else
    Body of else
```

Introduction to CPSC 233 James Tam

If, Else-If (2)

Example

```
if (gpa == 4)
{
    System.out.println("A");
}
else if (gpa == 3)
{
    System.out.println("B");
}
else if (gpa == 2)
{
    System.out.println("C");
}
```

If, Else-If (2)

```
else if (gpa == 1)
{
    System.out.println("D");
}
else
{
    System.out.println("Invalid gpa");
}
```

Alternative To Multiple Else-If's: Switch

```
Format
switch (variable name)
{
  case <integer value>:
  Body
  break;

  case <integer value>:
  Body
  break;
     :
  default:
  Body

}
```

1 The type of variable can be a byte, char, short, int or long

Alternative To Multiple Else-If's: Switch (2)

```
Format
switch (variable name)
{
  case '<character value>':
  Body
  break;

  case '<character value>':
  Body
  break;
     :
  default:
  Body

}
```

1 The type of variable can be a byte, char, short, int or long

Loops

Pascal Pre-test loops
- For-do
- While-do

Java Pre-test loops
- For
- While

Pascal Post-test loops
- Repeat-until

Java Post-test loops
- Do-while

While Loops

Format

```
While (Expression)
  Body
```

Example

```
int i = 1;
while (i <= 1000000)
{
    System.out.println("How much do I love thee?");
    System.out.println("Let me count the ways: ", + i);
    i = i + 1;
}
```

For Loops

Syntax:

```
for (initialization; Boolean expression; update control)
    Body
```

Example:

```
for (i = 1; i <= 1000000; i++)
{
        System.out.println("How much do I love thee?");
        System.out.println("Let me count the ways: ", + i);
}
```

Do-While Loops

Format:

```
do
    Body
while (Boolean expression);
```

Example:

```
    char ch = 'A';
    do
    {
        System.out.println(ch);
        ch++;
    }
    while (ch != 'K');
```

Ending Loops Early: Break

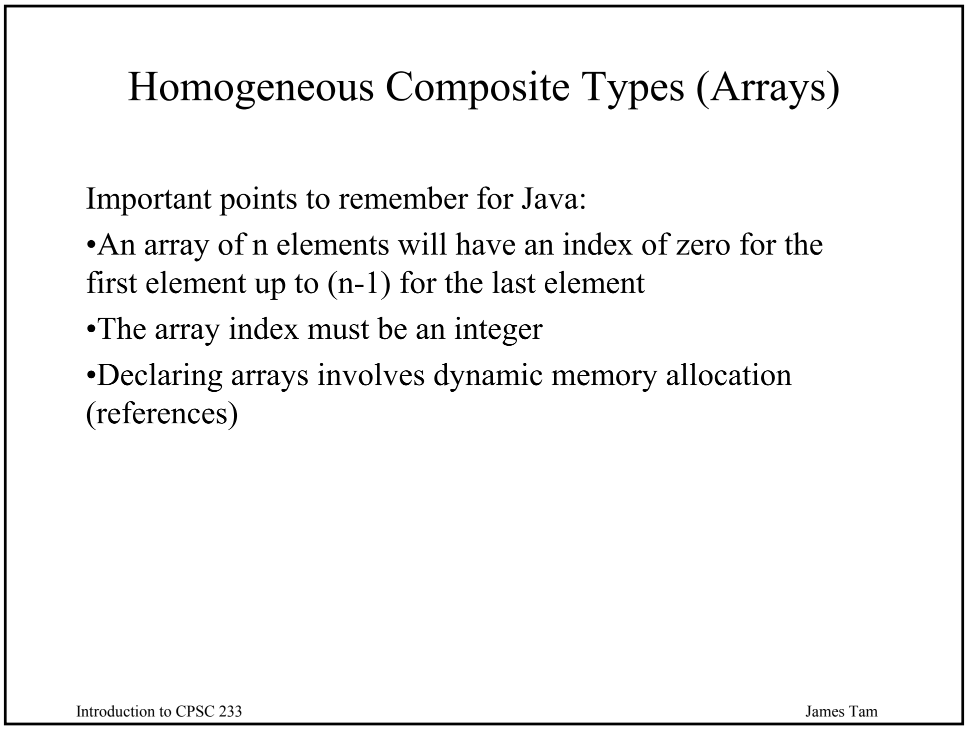
```
import tio.*;

class BreakExample
{
    public static void main (String [] argv)
    {
        char ch;
        while (true)
        {
          System.out.print("\tType 'Q' or 'q' to Quit program: ");
            ch = (char) Console.in.readChar();
            if ((ch == 'Q') || (ch == 'q'))
            {
                break;
            }
        }
    }
}
```

Introduction to CPSC 233 James Tam

Skipping An Iteration of A Loop: Continue

```
for (i = 1; i <= 10; i++)
{
  if (i % 2 == 0)
  {
      continue;
  }
  System.out.println("i=" + i);
}
```

Introduction to CPSC 233 James Tam



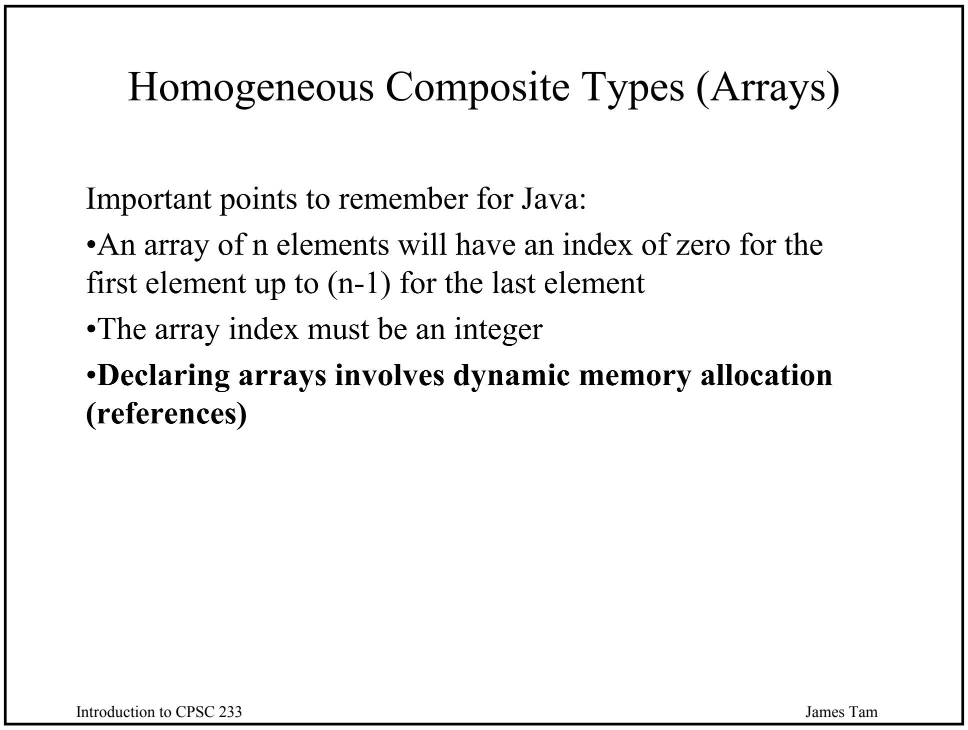
Homogeneous Composite Types (Arrays)

Important points to remember for Java:

•An array of n elements will have an index of zero for the first element up to (n-1) for the last element

•The array index must be an integer

•Declaring arrays involves dynamic memory allocation (references)



Homogeneous Composite Types (Arrays)

Important points to remember for Java:

•An array of n elements will have an index of zero for the first element up to (n-1) for the last element

•The array index must be an integer

•**Declaring arrays involves dynamic memory allocation (references)**

References

It is a pointer that cannot be dereferenced by the programmer

Automatically garbage collected when no longer needed

Introduction to CPSC 233 James Tam

References

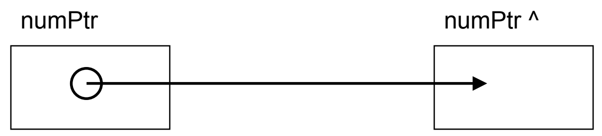
It is a pointer that cannot be dereferenced by the programmer

Automatically garbage collected when no longer needed

Introduction to CPSC 233 James Tam

Regular pointers (Pascal): Programmer Dereferencing

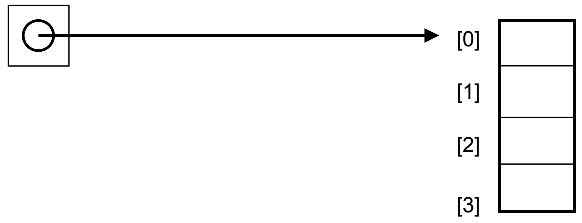
```
var
    numPtr : ^ integer;
begin
    new(numPtr);
```



References (Java): No Programmer Dereferencing

<reference type> <*reference name*> = new <reference type>;

```
final int SIZE = 4;
int [] arr = new int[SIZE];
```



References

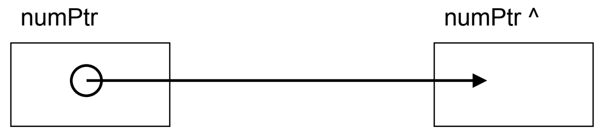
It is a pointer that cannot be dereferenced by the programmer

Automatically garbage collected when no longer needed

Introduction to CPSC 233 James Tam

Regular pointers (Pascal): Garbage Collection

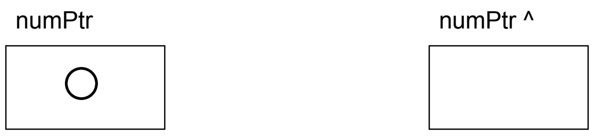
```
dispose (numPtr);
numPtr := NIL;
```



Introduction to CPSC 233 James Tam

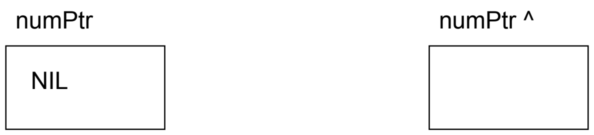
Regular pointers (Pascal): Garbage Collection

dispose (numPtr);
numPtr := NIL;



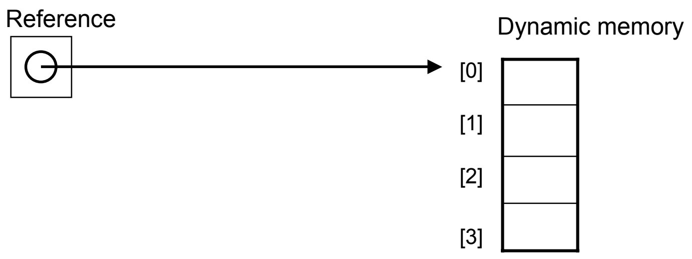
Regular pointers (Pascal): Garbage Collection

dispose (numPtr);
numPtr := NIL;



References (Java): Automatic Garbage Collection

Dynamically allocated memory is automatically freed up when it is no longer referenced



Determining Array Length

```
class Arr
{
   public static void main (String [] argv)
   {
      final int SIZE = 4;
      int [] arr = new int[SIZE];
      int i;
      for (i = 0; i < SIZE; i++)
      {
          arr[i] = i;
          System.out.println("Element " + arr[i] +"=" +i);
      }
      System.out.println(arr);
   }
}
```

Determining Array Length (2)

```
class Arr
{
   public static void main (String [] argv)
   {
      final int SIZE = 4;
      int [] arr = new int[SIZE];
      int i;
      for (i = 0; i < arr.length ; i++)
      {
          arr[i] = i;
          System.out.println("Element " + arr[i] +"=" +i);
      }
      System.out.println(arr);
   }
}
```

Declaring Arrays

Arrays in Java involve a reference to the array

Declaring array requires two steps:

1) Declaring a reference to the array
2) Allocating the memory for the array

Declaring A Reference To An Array

Syntax

<*type*> [] <array *name*>;

Example

```
int [] arr;
int [][] arr;
```

Allocating Memory For An Array

Syntax:

<*array name*> = new <*array type*> [<*no elements*>];

Example:

```
arr = new int[SIZE];
arr = new int[SIZE][SIZE];
```

Modularizing Programs: Methods (Definition)

Syntax:

```
<return type> name (<type1> <parameter1>, <type 2>
                   <parameter2>…<type n> <parameter n>)
{
   <Local variable declarations>
   <Statements>
   return (<return value>);
}
```

Modularizing Programs: Methods (Definition)

Example:

```
static double calculate (double principle, double interest,
                         double time)
{
    return(principle + (principle * interest * time));
}
```

Modularizing Programs: Methods (Definition)

Example:

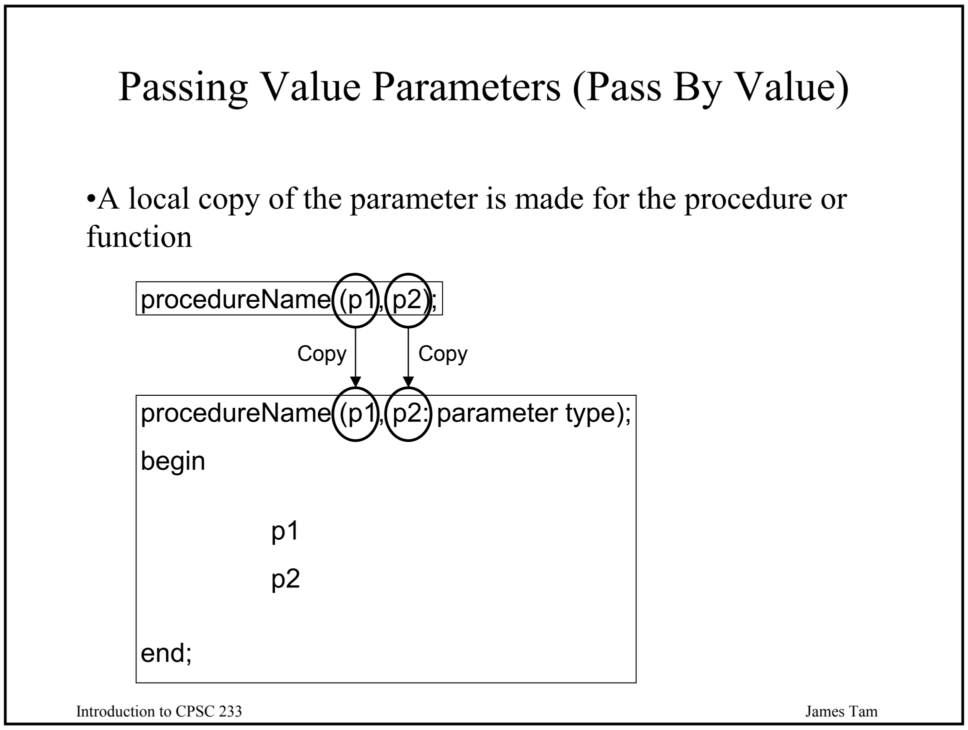
```
static double calculate (double principle, double interest,
                         double time)
{
    return(principle + (principle * interest * time));
}
```

Used for a 231-233 transition!

Parameter Passing: Review

Pascal

- Value parameters (pass by value)
- Variable parameters (pass by reference)

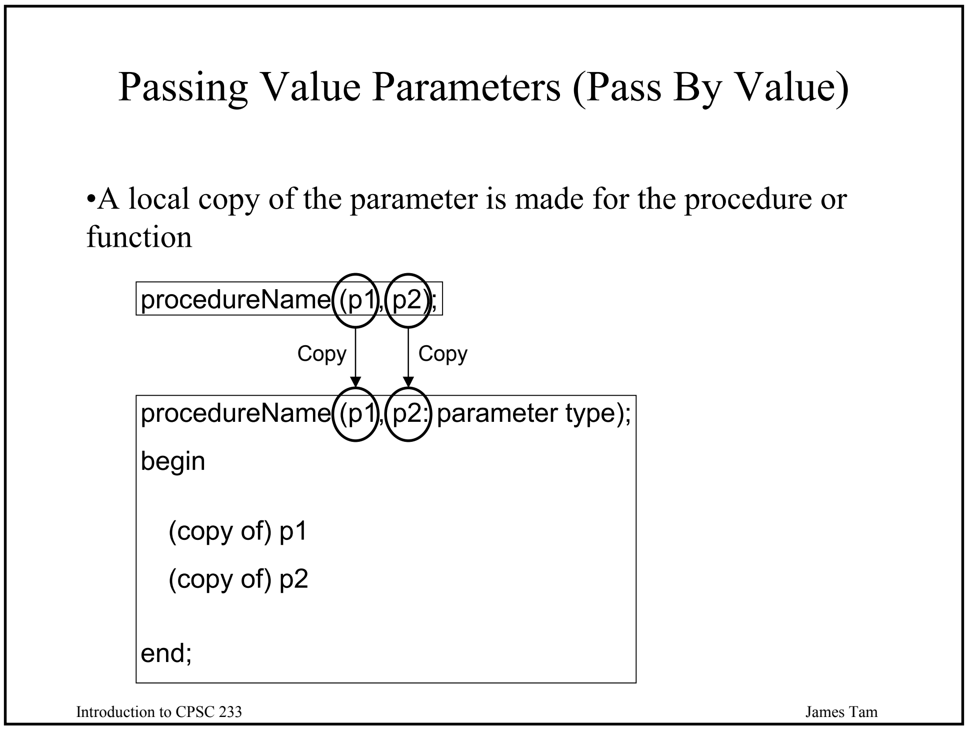


Passing Value Parameters (Pass By Value)

•A local copy of the parameter is made for the procedure or function

```
procedureName (p1), (p2);
```

```
procedureName (p1), (p2: parameter type);
begin

            p1
            p2

end;
```



Passing Value Parameters (Pass By Value)

•A local copy of the parameter is made for the procedure or function

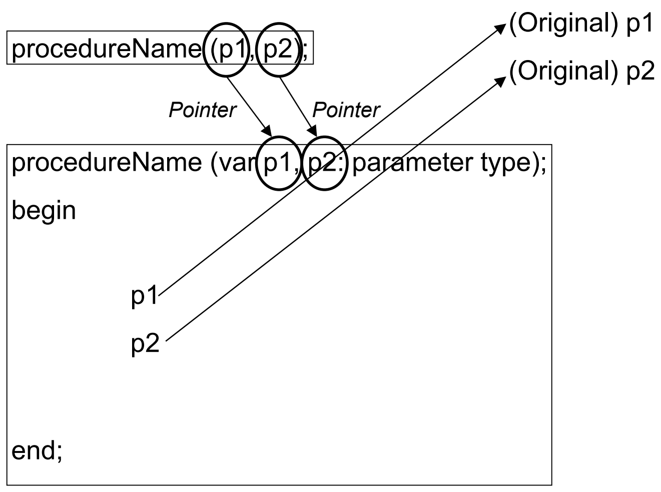
```
procedureName (p1), (p2);
```

```
procedureName (p1), (p2: parameter type);
begin

   (copy of) p1
   (copy of) p2

end;
```

Passing Variable Parameters (Pass By Reference)

- Changes made to the parameter inside the function or procedure will be retained outside of that module.
- This is done by passing a pointer to the parameter (refer to the parameter in the module and the pointer is automatically dereferenced in order to refer to the original parameter passed in)

Passing Variable Parameters (Pass By Reference)



Parameters To Methods In Java

Allowable parameters include:

- Simple type (boolean, char, integer etc.)
- Programmer defined types.

Simple types can only be passed by value in Java (copy passed in)

Composite types (arrays, objects) have a copy of a reference passed in (**not exactly pass by reference because the net effect is the same as with simple types**)

Method Return Types In Java

Any simple type (boolean, char, integer etc.)

Cases where no value is returned (void)

Programmer defined types (objects)

Passing Arrays As Parameters

Syntax:

<method return type> *<method name>* (<array type> [] <*name*>)

Example:

void fun (int [] arr)

void fun (int arr [])

Introduction to CPSC 233 James Tam

Methods In Java: Putting It All Together

See the example in Unix under:

/home/profs/tamj/233/examples/intro/Modules.java

Introduction to CPSC 233 James Tam

Some Common Mistakes

1. Semi-colons

   e.g., method definitions, last statements in a block of statements

2. Performing an assignment when you want to check for equity

   e.g., decision making constructs, loops

3. Comments

   i.e., (* and *) vs /* and */

   {}

Introduction to CPSC 233 James Tam