Java Programming

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

Important Dates

- Go over important dates.

Today's Lecture

- Chapter 6 (continued)

- Argument Promotion and Casting
- Enumerations
- Scope

Argument Promotion

- Argument Promotion – Converting an argument’s value (if possible) to the type the method expects in its corresponding parameter.

- For example:

```
double result;
result = Math.sqrt( 4 );
```

Sqrt expects a double

4 is an int but will be “promoted” to a double

```
double result;
result = Math.sqrt( 4 );
```

- The method declaration’s parameter list causes Java to convert the int value 4 to the double 4.0 before passing the value to the sqrt method.
- Can lead to compile errors if Java’s promotion rules are not satisfied.

Argument Promotion

Promotion rules – Specify which conversions are allowed (which ones can be done without losing data).

Is the following allowed by the promotion rules?

```
int x;
x = 7.0 / 2.0;  // Is this allowed???
```

Promotion in Expression

Promotion rules – Specify which conversions are allowed (which ones can be done without losing data).

Is the following allowed by the promotion rules? COMPILER ERROR.

```
int x;
```

$\underbrace{\text{~~x = 7.0 / 2.0;~~}}_{\text{This expression evaluates to a double}}$ // NOT ALLOWED!!!

Possible loss of data. Cannot assign a double to an int

Promotion in Expression

Promotion in Expression

```
double x;
x = 7.0 / 2 + 3;  // 2 and 3 are ints
```

Promote to double

- Each value in the expression is promoted to the “highest” type in the expression
- In the above example, 2 and 3 are promoted to doubles.
- The expression uses a temporary copy of each value.
- The types of the original values remain unchanged.

```
double x;
int y = 2;
x = 7.0 / y;

System.out.println(x);
System.out.println(y);
```

What will get printed???

Promotion in Expression

Promotion in Expression

```
double x;
int y = 2;
x = 7.0 / y;
```

A temp variable is created for y that is a double. y itself remains unchanged!!!

```
System.out.println(x);
System.out.println(y);
```

What will get printed???

ANSWER

```
3.5
2
```

Type	Valid Promotion
double	None
float	double
long	float or double
int	long, float, or double
char	int, long, float, or double
short	int, long, float, or double (but no char)
byte	short, int, long, float, or double (but no char)
boolean	None (boolean values are not numbers in Java)

<u>Type Descriptions</u>

byte	– 8-bit integer	float	– 32-bit floating point
short	– 16-bit integer	double	– 64-bit floating point
int	– 32-bit integer	char	– 16-bit Unicode character
long	– 64-bit integer	boolean	– 1-bit of information but the size is not defined.

Valid Promotions

- Compiler does not allow conversion when data may be lost.
- Use cast to get around this.
- A cast explicitly forces the conversion to occur (no compile error if you cast).
- For example:

```
int x;
double y;
x = (int) y;  // Casting the double as an int
```

The cast will remove the compile error. A temporary variable is created for y that is an int. y itself remains unchanged.

Casting

- Java enumeration is the next topic…

Enumeration

- Enumeration – A data type that is a set of constants represented by identifiers.

- Can only be defined in a top-level class or interface.

Create An Enum Type Definition

```
enum SizeType { SMALL, MEDIUM, LARGE };
```

- SizeType is a new data type. It can be used in a similar fashion as int, double, String, etc…

- The ***<u>ONLY</u>*** allowable values for this type are SMALL, MEDIUM, and LARGE.

Enumeration

Enumeration

Declare variable of the enum type

- Use the enumerated type as the variable data type

```
// s is a variable that has SizeType as its type
SizeType s;
```

Set enum variable to an enumerated value

- When using an enumerated value in a program you must prefix the enumerated value with the enumerated type.

```
// s gets the enumerated value MEDIUM
s = SizeType.MEDIUM;
```

Read Enumerated Value From File

- First, read the value in as a string.
- Next, use the valueOf method of the enumeration to convert the string to an enum type.

```
// Declare a String to store the value read from the file
enum SizeType { SMALL, MEDIUM, LARGE };
SizeType s;
String enumAsString;

// Read the value as a string from the file.
Scanner fileScanner = new Scanner(new FileReader("input.txt"));
enumAsString = fileScanner.next();

// Convert the SizeType to an enum
s = SizeType.valueOf(enumAsString);
```

Enumeration

Enumeration

- When putting a value of the enumerated type in a file you should only use the enumerated value.
- You should not prefix it with the enumerated type as you do inside the program when using a value of the enumerated type.
- Here is how you should write the enumerated value in an input file:

```
MEDIUM
```

- Scope of declarations is the next topic…

Scope of Declarations

- The scope of a declaration is the portion of the program that can refer to the declared entity by its name.

- Such an entity is said to be “in scope”.

- Basic scope rules:
  1. The scope of a **parameter** declaration is the body of the method in which the declaration appears.
  2. The scope of a **local-variable** declaration is from the point at which the declaration appears to the end of that block.
  3. The scope of a **local-variable that appears in the initialization section of a for statement’s header** is the body of the for statement and the other expressions in the header.
  4. A **method or field’s** scope is the entire body of the class. This enables non-static methods of a class to use the fields and other methods of the class.

Scope of Declarations

- Any block may contain variable declarations.
- If a local-variable or parameter in a method has the same name as a field of the class, the field is "hidden" until the block terminates execution (this is called **shadowing**).

```
public class X {
        private int a = 1;

        void S() {
                int a;
                a = 2; // Does not change member variable a!!!
        }
}
```

The local a variable shadows the member variable a

Shadowing

- Class member variable a is shadowed.
- Use this pointer to avoid shadowing.
- Prefix the class member variable with the this pointer then you can manipulate it even if it is shadowed in a method.

```
public class X {
        private int a = 1;

        void DoSomething() {
                int a;
                a = 2; // Does not change member variable a.
                this.a = 2; // Will change the member variable a!
        }
}
```

The local a variable shadows the member variable a

this pointer can be used to avoid the shadowing

Shadowing

Attendance

Take Attendance!!!

- **End of Slides**

End of Slides