Objects
(Chap. 2)

Variables and Constants

Our Temperature Code

We've looked at the overall structure of a Java application. Now we look at kinds of statements that are in the `main()` method.

```
/* Temperature.java converts Celsius
                          Fahrenheit.
                          nden
                          pt. 2002

                          creen & Keyboard classes
class Tempera             ends Object
{
  public static v         main(String [] args)
  {
    Screen theScreen = new Screen();
    theScreen.print("Welcome to the temperature converter!\n" +
           "Please enter the temperature in Celsius: ");

    Keyboard theKeyboard = new Keyboard();
    double celsius = theKeyboard.readDouble();

    double fahrenheit = ((9.0/5.0)*celsius) + 32;

    theScreen.print(celsius + " degrees Celsius is " +
           fahrenheit + " degrees Fahrenheit.\n" +
           "It's been a pleasure!\n");
  }
}
```


Statement Types

There are different types of statements in high-level programming languages:

- **Type declarations**
- **Expression statements**
- Control statements
- Input/Output (I/O) statements

We'll focus on the first two for now.

Types & Expressions

- In a Java program, any sequence of objects and operations that combine to produce a value is called an _____:
  - Objects are explicitly declared to be a certain _____
  - Operations are designed for a particular_____
- An example from our temperature problem:
- double fahrenheit = ((9.0/5.0)*celsius) + 32;

We will focus for now on Java objects.

Sec. 2.2

Data Types

- *All Java objects must have a type.*
- Java supports two categories of types:
  - _____ *types are the basic types*:
    - **`byte`**, **`short`**, **`int`**, **`long`**: integer values of various sizes (8, 16, 32, 64 bits)
    - **`float`**, **`double`**: real values (32, 64 bits)
    - **`boolean`**: logical (true/false) values (1 bit)
    - **`char`**: single characters (16 bits)

5

  - _____ types are built from other types:
    Examples:
    - **`String`**: for sequences of characters
    - **`Keyboard`**, **`Screen`**: associated with the standard input and output devices
    - Also called "class types"
    - Java 2 provides over 1600 reference types
- Primitive types are *known* to the compiler; reference types must be *explained* to it.
- _____ denotes the absence of any type.

6

Object Categories

- There are three kinds of objects:
- _____: unnamed objects having a value:
  - (0, -3, 2.5, 2.998e8, 'A', "Hello\n",...)
- _____: named objects whose values can change during program execution;
- _____: named objects whose values do not change during program execution;

7

Literals

Also for: `byte`, `short`, & `long`

Also for: `float`

- int literals are whole numbers:
  27, 0, 4, +4
- double literals are real numbers, and can be:
  - fixed-point: -0.333, 0.5, 1.414, ...
  - floating-point: 2.998e8, 0.2998e9, ...
- There are only two boolean literals:
  false, true
- char literals: single characters enclosed in single quotes
  'A', 'a', '9', '$', '?', ...
- String literals: character sequences enclosed in double quotes:
  "Hello", "Goodbye", "Goodbye\n",

8

Named Objects

- The name of an object is called an _____.
- Java identifiers must begin with a letter followed by zero or more letters, digits or underscores.
  - Valid: **`age`**, **`r2d2`**, **`myGPA`**, **`MAX_SCORE`**
  - Invalid: **`123go`**, **`coffee-time`**, **`sam's`**, **`$name`**
- *Identifiers cannot be Java reserved words* (e.g., names of primitive types, `import`, `class`)

9

Variable Declarations

- Variables are used to store value, but must first be _____. They can be either initialized or uninitialized in their declarations.
- Examples:
  - int age = 18;
  - double GPA = 3.25, credits;
  - char letterGrade = 'A';
  - bool ok, done = false;
- Pattern:
- type variableName [= expression];

10

SPECIAL HINT

- Pay close attention to patterns.
- Learn to read them:
  - Anything in normal font must be typed verbatim.
  - Anything in italics must be replaced with your own information.
  - Square brackets [...] indicate optional information.

11

***Note*:** In a variable declaration

`type variableName [ = expression ];`

→ ***`type`*** must be known to the compiler

→ ***`variableName`*** must be a valid identifier

→ ***`expression`*** is evaluated and assigned to ***`variableName`***'s memory location

→ If ***`= expression`*** is omitted, a default value is given (0, false, or null, depending on ***`type`***)

12

Assignment Statements

- The value of a variable can be changed using an assignment statement.
- Examples:
  - age = 19;
  - credits = hours * 3.0;
  - letterGrade = 'B';
  - done = true;
- Pattern: variableName = expression;

13

Constant Declarations

- Constants are used to represent a value with a meaningful name, and *must be initialized*.
- Examples:

```
final int MAX_SCORE = 100;
final double PI = 3.14159;
final char MIDDLE_INITIAL = 'A';
final String PROMPT = "Value: ";
```

- Pattern:

```
final type CONSTANT_NAME = expression;
```

14

Naming Conventions

- Variable names are all lowercase, with the first letter of each word after the first capitalized (e.g., **`lastName`**)
- Class names are like variable names except that the first letter is capitalized (e.g., **`LastName`**).
- Constant names are all uppercase, with multiple words separated by underscores (e.g., **`MAX_SCORE`**)

15

SPECIAL HINT

- Observe all programming conventions that we talk about.
- Conventions apply to all of the code you write, on quizzes and especially for labs and projects.
- You will not get this special hint again...

16

Part of the Picture: Data Representation

How literals of the primitive types are represented and stored in memory.

17

Representing Integers

Integers are often represented in the *twos-complement* format, where the high-order bit indicates the number's sign:

$2_{10} = 0000000000000010_2$
$1_{10} = 0000000000000001_2$
$0_{10} = 0000000000000000_2$
$-1_{10} = 1111111111111111_2$
$-2_{10} = 1111111111111110_2$

What's going on here and why?

These examples have 16 bits, but 32 or 64 are more common.

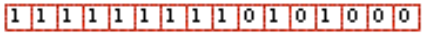
18

Two's-Complement

For nonnegative n:
Use ordinary base-two representation with leading (sign) bit 0

For negative n (–n):
1. Find w-bit base-2 representation of n
2. Complement each bit.
3. Add 1

Example: –88
1. 88 as a 16-bit base-two number
2. Complement this bit string
3. Add 1



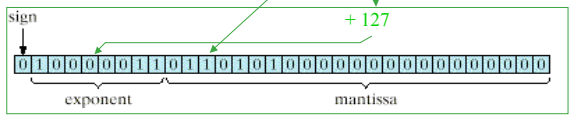
Shortcut for Step 3: Flip all bits from rightmost **0** to the end

19

Real Objects

Real values are often represented in 64 bits using the IEEE floating point standard:

Example: 22.625
Floating point form: $1.0110101_2 \times 2^4$

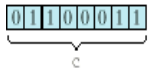


20

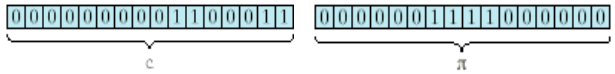
Character Objects

Store numeric codes (ASCII and Unicode are standard

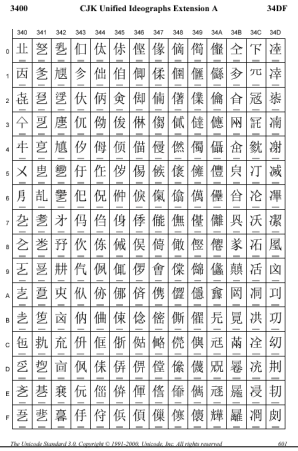
ASCII uses 1 byte (8 bits) per character, allowing for $2^8 = 255$ characters



Java uses Unicode, which uses 2 bytes (16 bits) per character, allowing for $2^{16} = 65536$ characters (see examples on p. 68).



UNICODE supports a number of different character types (see www.unicode.org)



Representing Booleans

- Only two possible values
  `true` and **`false`**
- Only need two possible numbers, 0 and 1
- Single bit is all that is needed

Sec. 2.3

Some Basic Program Features

Using _____, we can build new types to model real world objects that can't be represented using available types.

Pattern:

```
class ClassName extends ExistingClassName
{
   // attributes (variables & constants)
   // and behaviors (methods)
}
```

- ***`ClassName`*** is the name of a new reference type
- ***`ExistingClassName`*** is any class name known to the compiler
- { and } mark the boundaries of the declaration

An _____ is a program entity whose type is a _____

Importing Packages

Related classes can be grouped together into a container called a "package." A program specifies in what package to find a desired class

- *Fully-qualified name* of a class:
  PackageName.ClassName
  PackageName1.PackageName2.ClassName
- Using **`import PackageName;`** makes it possible to omit the prefixes and dot notation.
- Pattern:
  `import PackageName.*;` or
  `import PackageName.ClassName;`
  where ***`ClassName`*** is any class stored in ***`PackageName`***

25

Using Methods

- We *call*, *invoke*, or *send a message to* a method of an existing object, by using dot notation.

  Pattern:

  `objectName.methodName(arguments)`
- Example,

  `theScreen.print(" ... ");`
  - **`theScreen`** is the object
  - **`print( )`** is the method being called

26

Sec. 2.4

Java Documentation – API

- Java designers have provided over 1600 classes
  - Called the Java Application Programmer's Interface or API
  - Each class provides variety of useful methods
  - Classes grouped into packages
- To find a needed package or class, use the hypertext-based documentation system:

`http://java.sun.com/j2se/1.4.1/docs/api`

This is an important reference source and you should learn to use it effectively

27