

COMPUTER PROGRAMMING IN JAVA

COLUMBIA UNIVERSITY HIGH SCHOOL SCIENCE HONORS PROGRAM

2006 Feb 03 Sat

1. Review

- What is computer science?
 - Essentially, the study of algorithms.
- What is an algorithm?
 - A step-by-step, unambiguous procedure for completing a well defined task.
 - Examples
 - Euclid's algorithm for finding the greatest common denominator
 - Starting a car
 - Driving a manual transmission car
 - Sorting a deck of cards
- How Java works
 - source file
 - text, human-readable, in the Java language
 - class file
 - binary, machine-readable, for the JVM

2. Data types & Variables

- boolean
 - true or false
- “Integers”
 - short
 - 16 bits
 - Smallest data type for storing whole numbers
 - -32,768 to 32,767
 - int
 - 32 bits
 - Standard integer data type
 - long
 - 64 bits (big!)
 - Big!
 - -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (+/- 9 quintillion)
- “Floating point numbers”
 - float
 - 32 bits
 - double
 - 64 bits
 - Standard floating point data type

- “Strings”
 - char
 - 16 bits
 - Unicode
 - String
 - An array of characters
 - Examples
 - “hello”
 - “world”
 - Also Unicode
- “Arrays”
 - int[], boolean[], etc
 - 0-indexed!
- What is a variable?

Think of the memory layout: it's the human-readable name of a location in memory. Or you can think of it as a box that holds a value. The value it holds is determined by its data type.

To create a variable, we first have to declare it. Note that we can declare a variable more than once in the same scope.

- How to assign?

With the assignment operator “=”. E.g.: int i = 5, boolean b = false. Invalid assignments: String s = 1, double d = 'z'. This is called “type checking”. Java is “strongly typed”.

3. Comparison Operators

Now that we know the data types, what can we do with them? How about basic true/false statements?

- ==
 - is equal to (note that it's **TWO** equals signs, not one!)
- !=
 - is not equal to
- >
 - is greater than
- >=
 - is greater than or equal to
- <
 - is less than
- <=
 - is less than or equal to

- &&
 - logical AND
- ||
 - logical OR
- !
 - logical NOT

Important: Strings use .equals()!

4. Arithmetic Operators

Two types:

- Binary
 - +
 - -
 - *
 - /
 - %
 - Promotion rules
 - It's okay to make something “bigger”: widening promotions
 - int -> float
 - short -> long
 - It's bad to make something “smaller”: narrowing promotions
 - double -> int
 - long -> short
 - String promotions
 - int, short, long, float, double, char -> String
- Unary
 - +
 - unary plus
 - -
 - unary minus
 - ++
 - increment
 - post versus pre
 - --
 - decrement
 - post versus pre

5. Control Flow

Remember the definition of an algorithm: a step-by-step series of unambiguous instructions. Part of the algorithm might involve doing something conditionally, or doing another thing many times. The features of a programming language that allow us to do

this are collectively called “control flow structures”.

- if/else if/else

```
if ( <boolean> )
{
    ...
}
else if ( <boolean> )
{
    ...
}
else if ( <boolean> )
{
    ...
}
...
else
{
    ...
}
```

- for loop

```
for ( <variable initialization>; <boolean>; <step assignment> )
{
    ...
}
```

- while loop

```
while ( <boolean> )
{
    ...
}
```

Note that a semicolon delimits a “statement”.

6. Random numbers

```
import java.util.Random;

Random random = new Random();
int x = rand.nextInt(100);
```

Note the range! [0, 100)

7. User I/O & Driver

- Scanner class.
 - The simplest way to read user input from the command line
 - Must first import!
- System.out.print, System.out.println
 - One prints without a newline, the other with a newline
- public static final void main(String[] args)
 - The “driver”
 - This is where program execution begins

8. Assignments

Configure your Windows environment. We need:

1. A text editor; notepad will do.
2. A command prompt
3. The commands javac and java.

Easy

- Modify the number guessing game
 - Description
 - This is a number guessing game. The computer randomly picks a number between 1 and 10 (inclusive), and the player has to guess the number. Each time the player gets it wrong, they are told whether their guess was too high or too low.
 - Task
 - The code has already been written for you, implementing the rules as described above. The source file is called “GuessingGame.java”.
 - Modify the guessing game to only allow 10 guesses before the player loses and the computer reveals the answer.
- Time conversion
 - Description
 - Write one program that asks the user to enter a number of seconds (as an integer), and then prints the equivalent amount of time as a combination of hours, minutes, and seconds.
 - Now write another program that does the reverse: Given the number of hours, minutes, and seconds, print the equivalent number of seconds in total.
 - Task
 - Two source files are provided for you: “ToSeconds.java” and “FromSeconds.java”. Both of them provide the code to prompt the user for input.

Medium

- Rock, Paper, Scissors
 - Description
 - Write a program that lets a user play “Rock, Paper, Scissors” against the computer. The program should ask the user to choose one of the three choices, and then the computer randomly picks one (without knowing what the user has chosen). Rock beats scissors, scissors beats paper, paper beats rock. The program should say who wins, and then keep playing until someone (the user or the computer) has won 10 round. The computer needs to keep track of the current score and should also display it before each round.
 - Task
 - A basic template source file has been written for you called “RockPaperScissors.java”.
- Implement an algorithm
 - Description
 - Consider the following algorithm:
 1. Pick a positive integer n.
 2. Generate n random integers, each between 0 and n, and place them all into an array.
 3. Iterate through the array, adding up each element in the follow manner:
 - a) If the number is even, add it to the total
 - b) If the number is odd, subtract it from the total
 - Task
 - Implement the algorithm described above.
 - Make sure to print out the array of random numbers you generate
 - Print the final result.
 - A basic template source file has been written for you called “Algorithm.java”.

Hard

- The over-under dice game
 - Description
 - Consider the following dice game played between a human player and a computer player. Each player simultaneously rolls two dice and observes only their own roll. The computer player then declares an over-under value (an integer between 4 and 24) based on an estimate of the sum of the face values over all four dice. The player then guesses whether the actual sum is over or under the computer's stated value. If the sum of the four dice is exactly the number the computer picks, then the round is a tie. Otherwise if the player chooses correctly, s/he is awarded a point. If the player chooses incorrectly, the computer is awarded a point. The first player to 10 points wins.
 - For example:
 - The dice (all four) are rolled.
The computer observes it has rolled 4 and 5 (but the human player doesn't

know this).

The computer estimates the line at 16 (without knowing what the human player has).

The human player observes s/he has rolled 5 and a 6.

The human declares over.

Since $11 + 9 > 16$, the human player earns a point.

The game continues.

In your program, the computer will estimate the over-under line by randomly choosing a number between 6 and 8 and adding it to its own total.

- Task
 - Implement the over-under game as described above.
- Election simulation
 - Description
 - Imagine you are working as a political campaign consultant and you wish to write a computer program to model an election. Your idea is to model the behavior of each individual voter as the product of two random events: (A) They show up to vote, (B) They vote for your candidate. This means that each voter will have two numbers between 0 and 1 associated with them: (A) P-attend = the probability that they show up, (B) P-vote = the probability they vote for your boss.

We can assume that this is a two-candidate race so if P-vote is the probability that a voter will vote for your boss then the probability that they vote for the other guy is $(1 - P\text{-vote})$. Based on values of P-attend and P-vote we may group the voting population into three categories:

- a. (35% of the voters) Your base (P-attend=0.5, P-vote=0.8)
- b. (35% of the voters) Your opponent's base (P-attend=0.55, P-vote=0.2)
- c. (30% of the voters) Swing vote (P-attend=0.4, P-vote=0.5)

There are exactly 5,000 registered voters in your district.

- Task
 - Your job is to write an application that simulates such an election to determine an estimate of the probability that your candidate wins. To do this, have your application simulate each of the 5,000 voters. Count the votes your candidate receives, and print the winner.