Java Continued and Nim

Instructors: Sam McCauley and Dan Barowy
February 9, 2022
Control Flow and Loops
Two versions of a loop

Random(rng = new Random();
int flip = rng.nextInt(2);
int count = 1;
while (flip == 0) {
    // count flips until "heads"
    flip = rng.nextInt(2);
    count++;
}

Random(rng = new Random();
int flip = rng.nextInt(2);
int count = 1;
for(int count=1; flip==0; count++){
    flip = rng.nextInt(2);
}

One more version of the loop

```java
Random rng = new Random();
int flip = rng.nextInt(2), count = 1;
while (flip == 0) {
    // count flips until "heads"
    flip = rng.nextInt(2);
    count++;
}
```

```java
int flip, count = 0;
do {
    //count flips until "heads"
    flip = rng.nextInt(2);
    count++;
} while (flip == 0);
```
Control Structures

- Select next statement to execute based on value of a boolean expression. Two flavors:
  - Looping structures: while, do/while, for
    - Repeatedly execute same statement (block)
  - Branching structures: if, if/else, switch
    - Select one of several possible statements (blocks)
    - Special: break/continue: exit a looping structure
      - break: exits loop completely
      - continue: proceeds to next iteration of loop
      - break and continue are *to be avoided* unless it greatly simplifies or clarifies your code
If/else

```java
if (x > 0)  // There is exactly 1 "if" clause
    y = 1 / x;
else if (x < 0) {  // 0 or more "else if" clauses
    x = -x;
    y = 1 / x;
}
else  // at most 1 "else" clause
    System.out.println("Can’t divide by 0!");
```
int x = myCard.getSuit(); // a fictional method
//0 is spades; 1 is diamonds; 2 is hearts; 3 is clubs
switch (x) {
    case 1: case 2:
        System.out.println("Your card is red");
        break;
    case 0: case 3:
        System.out.println("Your card is black");
        break;
    default:
        System.out.println("Illegal suit code!");
        break;
}
For & for-each

Here’s a typical **for** loop example

```java
int[] grades = { 100, 78, 92, 87, 89, 90 }; int sum = 0;
for( int i = 0; i < grades.length; i++ )
    sum += grades[i];
```

This **for** construct is equivalent to

```java
int[] grades = { 100, 78, 92, 87, 89, 90 }; int sum = 0;
int i = 0;
while ( i < grades.length ) {
    sum += grades[i];
    i++;
}
```
For & for-each

Here’s a typical `for` loop example

```java
int[] grades = { 100, 78, 92, 87, 89, 90 };
int sum = 0;
for (int i = 0; i < grades.length; i++)
    sum += grades[i];
```

Can also write (`for-each` construct; will see more later)

```java
int[] grades = { 100, 78, 92, 87, 89, 90 };
int sum = 0;
for (int g : grades )
    sum += g;
```
The body of a **while** loop may not ever be executed.

The body of a **do – while** loop always executes at least once.

**For** loops are typically used when number of iterations desired is known in advance. E.g.
- Execute loop exactly 100 times
- Execute loop for each element of an array

The **for-each** construct is often used to access array (and other collection type) values when *no updating* of the array is required.
- We’ll explore this construct more later in the course.
Methods in Java
Why methods?

- Used to group together code
  - Well-organized code is often superior to well-documented, poorly-organized code.

- A method should do one task

- Methods allow us to reuse code as well as use techniques like recursion.
Creating and using methods

• We can create a method as follows:

```java
public static int getSum(int a, int b){
    return a+b;
}
```

• (We’ll talk about public and static next week.)

• We can call a method as follows (this prints the sum of 3 and x):

```java
System.out.println("The sum is " + getSum(3, x));
```
The String & Scanner Classes
The String Class

- String is not a primitive type in Java, it is a *class type*
- However, Java provides language level support for Strings
  - String literals: “Bob was here!”, “-11.3”, “A”, “ ”
- A single character can be accessed using charAt()
  - As with arrays, indexing starts at position 0
  - String s = “computer”;
  - char c = s.charAt(5); // c gets value ‘t’
  - c = “oops”.charAt(4); // run-time error!
- String provides a length method
  - int len = s.length(); // len gets value 8
  - len = “ “.length(); // len gets value 0
Scanner class

• A way to get interactive input from a user!

• Not built-in; need to import in order to use:
  • import java.util.Scanner;

• First, instantiate a Scanner:
  • Scanner sc = new Scanner(System.in);

• Then, can use it to read in lines of text:
  • System.out.println("Enter your name:");
  • String name = sc.nextLine();

• Let’s look at an example: GuessNumber.java
Object Oriented Programming
The Plan

• I want to *briefly mention* objects today

• We’ll be filling in details starting on Friday!

• OK if you don’t completely get it—just some foundational concepts and vocab
Objects

• **Primitive types** are just data in Java: an int just stores a number; a char just stores a character

• And nothing else!

• An **object** is fancier. It may store extra data, or multiple pieces of data. It may even store some *methods* along with the data

• For example:
  
  • An array doesn’t just store the data—it also stores the `length`
  
  • A String has a `.length()` method
  
  • A Random object has a `.nextInt()` method, and stores data to help generate random numbers
Objects and Primitive Types

- Objects need to be instantiated with `new`

- You’ll be making your own types of objects very soon! But for lab 1, only need to use the kinds of objects we’ve already discussed (String, Scanner, Random, etc.)
Nim
This section

- Let’s talk about a game
- And then code it up!

- Goals:
  - Java practice and...
  - Maybe some useful ideas for lab 1?
At the game’s start, there are one or more piles of matchsticks.

- Players take turns.
- The player whose turn it is must choose one pile and remove one or more matchsticks from that pile.
- The player who cannot remove a matchstick loses (i.e., the winner removes the very last matchstick from the gameboard).

Let’s play a quick game of Nim
How can we code this up?

• How should we store the piles?
• How do we create the board?
• What is a legal move?
• How do we have it play the game?
Let’s Code up Nim!
Design Documents

- Example on website
- Idea: read through the lab
- Describe how you will implement it
  - How will you store the data?
  - What methods will you use?
  - Etc.
- We’ll be collecting them in lab (so remember to bring them)!