CS18000: Problem Solving and Object-Oriented Programming

Repetition

Video 1
Repetition Concept

Repetition

Concepts

Indefinite Iteration

while Loops and Examples

What's Missing? Lots of Data

- You know how to make a decision and execute one statement or another
- Problems up to now involved working on a fixed set of input or small number of values
- Next power up: perform repetitive actions

if statement vs. while statement

```
if (boolean-expression)
   then-statement;
next-statement;

while (boolean-expression)
   loop-statement;
next-statement;
```

Repetition Concept

- Repetition broken into two parts
  - Body of code that gets repeatedly executed
  - Condition (boolean) to determine when to stop
- How to construct the body so that it does something different/useful each time it is run?
- The state of the computation must change with each iteration (otherwise nothing is done)

Two Forms of Iteration

- Indefinite: loop until “done”; no advance knowledge of how many iterations will be required
- Definite: loop a given number of times; used when the iterations are controlled by a counter or size or limit

Java Repetition Constructs

- while loop
  - Check a boolean condition
  - If true, execute a block of statements
  - Repeat
- do-while loop
  - Execute a block of statements
  - If a boolean condition is true, repeat
- for loop

Video 2
Simple Repetition Examples

Problem: Odd or Even

- Write a program that reads integers from standard input; for each integer print a message indicating “odd” or “even”
- Stop reading when no more integers
- Scanner method `hasNextInt()` returns true if there is another integer available, else returns false
- Scanner method `hasNextInt()` returns false at EOF; also returns false if something other than an integer found

Solution: Odd or Even

```
import java.util.Scanner;

public class OddOrEven {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int number;
        while (in.hasNextInt()) {
            number = in.nextInt();
            if (number % 2 == 0)
                System.out.printf("%d is even\n", number);
            else
                System.out.printf("%d is odd\n", number);
        }
    }
}
```

Solution: Odd or Even

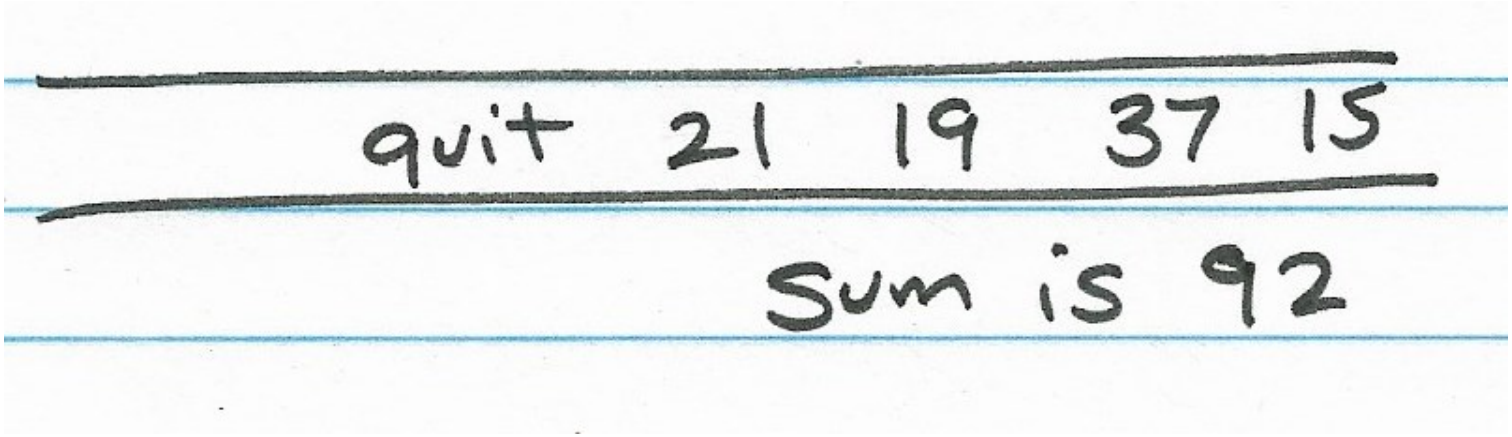
Problem: Summer

- Read a sequence of integers from the standard input and compute their sum
- Two problems:
  - How do we know when we're done?
  - How do we accumulate the sum?
- Also count the number of values read

Solution: Summer

```
import java.util.Scanner;

public class Summer {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        int number;  // number that is input
        int sum = 0; // sum of values
        int c = 0; // how many values read

        while (in.hasNextInt()) {
            number = in.nextInt();
            c = c + 1;
            sum = sum + number;
        }

        System.out.printf("sum of %d values is %d\n", c, sum);
    }
}
```

Solution: Summer



Video 3
Palindrome Checker with while Loop

Repetition

Indefinite Iteration Examples

Definite Iteration

The while Loop

```
while (boolean-expression) {
    statements;
}
```

1. Test boolean-expression first
2. If true, do statements in body
3. Repeat from 1.

Problem: Palindrome

- Write a method in class Palindrome

```
boolean isPalindrome(String s)
```

- to test if s is a palindrome, a string that reads the same backwards as forwards
- Approach 1: Use a while loop

String s = new String (“racecar”);

```
r a c e c a r
0 1 2 3 4 5 6
```

Strategy: Palindrome

- Compare first and last characters; differ? False
- Strip off first and last characters
- Repeat until length < 2; return true
- Test input:
  - “level” (true)
  - “racecar” (true)
  - “henway” (false)
  - “x”, “aba”, “abba” (all true)
  - “” (empty string (true))
  - null (null value (true))

Solution: Palindrome

```
public class Palindrome {
    boolean isPalindrome(String s) {
        if (s == null || s.length() <= 1)
            return true;

        while (s.length() > 1) {
            char first = s.charAt(0);
            char last = s.charAt(s.length() - 1);
            if (first != last)
                return false;
            s = s.substring(1, s.length() - 1);
        }
        return true;
    }
}
```

Solution: Palindrome

```
racecar
0123456

aceca
01234

cec
012

e
0
```

Solution: PalindromeTest

```
import junit.framework.TestCase;

public class PalindromeTest extends TestCase {
    public void testIsPalindrome() {
        Palindrome p = new Palindrome();
        assertEquals(true, p.isPalindrome(""));
        assertEquals(true, p.isPalindrome(null));
        assertEquals(true, p.isPalindrome("x"));
        assertEquals(true, p.isPalindrome("xx"));
        assertEquals(false, p.isPalindrome("xy"));
        assertEquals(true, p.isPalindrome("level"));
        assertEquals(false, p.isPalindrome("henway"));
        assertEquals(true, p.isPalindrome("racecar"));
    }
}
```

Video 4
Continue, Break, Increment, Decrement

Problem: Reverse

- Add a method

```
String reverse(String s)
```

- to Palindrome to reverse a String

- isPalindrome could simply be…

```
boolean isPalindrome(String s) {
    return s.equals(reverse(s));
}
```

Continue statement

```
        while (in.hasNext()) {
            word = in.next();
            if (word.length() != 4)
                count=count+1;
        }
// is equivalent to….
        while (in.hasNext()) {
            word = in.next();
            if (word.length() == 4)
                continue;
            else
                count=count+1;
        }
```

Break statement

```
while (in.hasNext()) {
    word = in.next();
    if (word.length() == 4)
        break;
    else
        count=count+1;
}
```

Pro Tip: Compound Assignment

- Common assignment statements:

```
x = x + y;
a = a – b;
s = s + "\n"; // s is a string
```

- Java provides shortcut to save keystrokes:

```
x += y;
a -= b;
s += "\n";
```

- Available for all (or most) binary operators

Pro Tip 2: Increment/Decrement Operators

- A refinement for an even more common case:

```
x = x + 1;
a = a – 1;
```

- Java provides even more keystroke savings:

```
x++;
a--;
```

- Also:

```
++x;
--a;
```

Post- and Pre- Increment/Decrement

- x++ increments x by one, but the expression value is the original x

```
int x = 0;
System.out.println(x++);
System.out.println(x);
```

- Prints 0, then 1
- ++x increments x by one, and its value is the new x

```
int x = 0;
System.out.println(++x);
System.out.println(x);
```

- Prints 1, then 1
- x++ and x-- are very common idioms (cf. C++)
- Life becomes messy if an expression contains multiple pre- and post- increment and decrement operators

Video 1
for Loop

Problem: WhileDefinite

- Problem: Print “hello” 10 times using a while loop
- Illustrates using a while loop to implement a definite iteration

Solution: WhileDefinite

```
public class WhileDefinite {
    public static void main(String[] args) {
        int n = 0;
        while (n < 10) {
            System.out.printf("hello (#%d)\n", n);
            n++;
        }
    }
}
```

Solution: WhileDefinite

```
hello (#0)
hello (#1)
hello (#2)
hello (#3)
hello (#4)
hello (#5)
hello (#6)
hello (#7)
hello (#8)
hello (#9)
```

The Loop Parts

```
public class WhileDefinite {
    public static void main(String[] args) {
        int n = 0;
        while (n < 10) {
            System.out.printf("hello (#%d)\n", n);
            n++;
        }
    }
}
```

e1. Initialize

e2. Test

e3. Update

```
        for (int n = 0; n < 10; n++)
            System.out.printf("hello (#%d)\n", n);
```

Definite Iteration: for loop

- Very general form:

```
for (e1; e2; e3) { statements; }
```

- Sequence of actions:
  1. Evaluate expression e1 (once only).
  2. Evaluate e2. If true, execute statement body.
  3. Evaluate e3.
  4. Return to step 2.

Common Practices

- To loop n times, go from 0 to n-1

```
for (int i = 0; i < n; i++) { statements; }
```

- Works well for strings (and arrays): 0-based
- To print the characters in a String s:

```
String s = "hello there world";
for (int i = 0; i < s.length(); i++)
    System.out.printf(
          "s.charAt(%d) = '%c'\n",
          i, s.charAt(i));
```

Video 2
do-while Loop and Palindrome Checker with for Loop

Repetition

do-while Loop

for Loop

Nested Loops and Other Examples

The do-while Loop

```
do {
    statements;
} while (boolean-expression);
```

1. Execute statements in body
2. Test boolean-expression
3. If true, repeat from 1

Problem: Prompting the User

- Write a program, Prompter, that prompts the user for an even number
- Continue prompting until an even number is provided

- Show alternate implementation using standard while loop with sentinel

Solution 1: Prompter1

```
import java.util.Scanner;

public class Prompter1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        // Prompt for an even number using do-while...
        int n;
        do {
            System.out.printf("Please enter an even number: ");
            n = in.nextInt();
        } while (n % 2 == 1);

        System.out.printf("Thank you for entering the even number %d\n", n);
    }
}
```

Solution 2: Prompter2

```
import java.util.Scanner;

public class Prompter2 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = 0;

        // Prompt for an even number with while, using sentinel...
        boolean noEvenYet = true;
        while (noEvenYet) {
            System.out.printf("Please enter an even number: ");
            n = in.nextInt();
            if (n % 2 == 0)
                noEvenYet = false;
         }

        System.out.printf("Thank you for entering the even number %d\n", n);
    }
}
```

Problem: Palindrome (Redone)

- Write a method in class Palindrome

```
boolean isPalindrome(String s)
```

- to test if s is a palindrome (reads the same backwards as forward)
- Approach 2: Use a for loop

Palindrome: Strategy

- Iterate through the first half of the string
- Compare current character to corresponding character at other end of string
- Consider table of indexes i and j for “racecar”
- Find pattern, generalize to solution

```
i j
0 6
1 5
2 4
```

Palindrome: Solution

```
 boolean isPalindrome(String s) {
    if (s == null)
        return true;

    for (int i = 0; i < s.length() / 2; i++)
        if (s.charAt(i) != s.charAt(s.length() - 1 - i))
            return false;
    return true;
}
```

Video 3
Common Mistakes

Common Mistakes

- Infinite loop
- Almost infinite loop
- Fencepost errors
- Skipped loops
- Misplaced semicolons

Infinite Loop

```
public class InfiniteLoop {
    public static void main(String[] args) {
        int n = 1;
        while (n < 100) {
            System.out.printf("n = %d\n", n);
            // forgot to increment n
        }
    }
}
```

Almost Infinite Loop

```
public class AlmostInfiniteLoop {
    public static void main(String[] args) {
        // count down to blast off
        for (int i = 10; i > 0; i++)
            System.out.printf("%d\n", i);
        System.out.printf("BLAST OFF!\n");
    }
}
```

Fencepost Error

```
public class FencePostError {
    public static void main(String[] args) {
        for (int i = 0; i <= 5; i++)
            System.out.printf(
            "print this line 5 times (%d)\n", i);
    }
}
```

Skipped Loop

```
import java.util.Scanner;

public class SkippedLoop {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int number = 0;
        int sum = 0;

        // read ints from user until zero, then print sum
        while (number > 0) {
            sum += number;
            number = in.nextInt();
        }
        System.out.printf("sum = %d\n", sum);
    }
}
```

Fixing the Skipped Loop

```
import java.util.Scanner;

public class SkippedLoop {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int number = 0;
        int sum = 0;
        number = in.nextInt(); // priming read
        // read ints from user until zero, then print sum
        while (number > 0) {
            sum += number;
            number = in.nextInt();
        }
        System.out.printf("sum = %d\n", sum);
    }
}
```

Misplaced Semicolon

```
public class MisplacedSemicolon {
    public static void main(String[] args) {
        int i = 10;
        while (--i >= 0); {
            System.out.printf("message #%d\n", i);
        }
    }
}
```

Video 4
Nested Loops and convertToBinary Example

Nested Loops

- Just like you can nest if statements
- You can also nest loops
- Inner loop is run completely for each iteration of outer loop:

```
public class Nested {
    public static void main(String[] args) {
        for (int i = 0; i < 5; i++)
            for (int j = 0; j < 5; j++)
                System.out.printf("i = %d, j = %d\n",
                                  i,      j);
    }
}
```

Nested Loops

```
i = 0, j = 0
i = 0, j = 1
i = 0, j = 2
i = 0, j = 3
i = 0, j = 4
i = 1, j = 0
i = 1, j = 1
i = 1, j = 2
i = 1, j = 3
i = 1, j = 4
i = 2, j = 0
i = 2, j = 1
i = 2, j = 2
i = 2, j = 3
i = 2, j = 4
```

```
i = 3, j = 0
i = 3, j = 1
i = 3, j = 2
i = 3, j = 3
i = 3, j = 4
i = 4, j = 0
i = 4, j = 1
i = 4, j = 2
i = 4, j = 3
i = 4, j = 4
```

Problem: Draw Divisor Pattern

- Print an nxn table
- The entry at row i and column j has an * if i divides j or j divides I
- Example for n == 5

	1	2	3	4	5
1	*	*	*	*	*
2	*	*		*	
3	*		*		
4	*	*		*	
5	*				*

Solution: Draw Divisor Pattern

```
public class DivisorPattern {
    public static void main(String[] args) {
        int n = 5;

        System.out.printf("  ");
        for (int i = 1; i <= n; i++)
            System.out.printf("%3d", i);
        System.out.printf("\n");

        for (int i = 1; i <= n; i++) {
            System.out.printf("%3d", i);
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0)
                    System.out.printf("  *");
                else
                    System.out.printf("   ");
            }
            System.out.printf("\n");
        }

    }
}
```

Problem: convertToBinary

- Create a class Converter with method

```
String convertToBinary(int n)
```

- that converts n to binary equivalent, as a String of 0s and 1s
- Use while loop

Solution: convertToBinary

```
public class Converter {
    String convertToBinary(int n) {
        String result = "";

        // handle special cases...
        if (n < 0)
            return null; // failure
        if (n == 0)
            return "0";

        // loop while n > 0, accumulating a bit and dividing by 2...
        while (n > 0) {
            if (n % 2 == 0)
                result = "0" + result;
            else
                result = "1" + result;
            n = n / 2;
        }
        return result;
    }
}
```