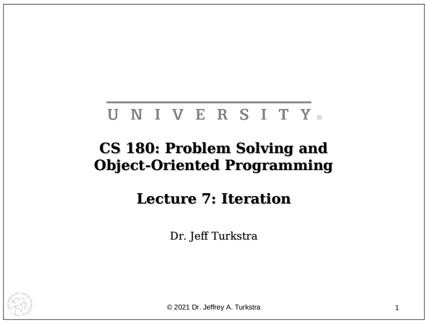
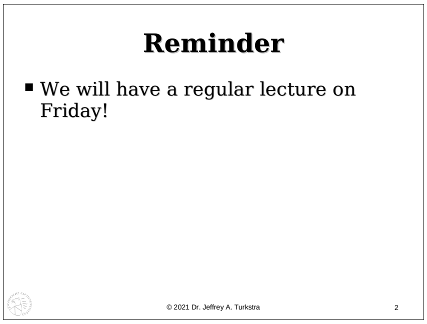

UNIVERSITY®

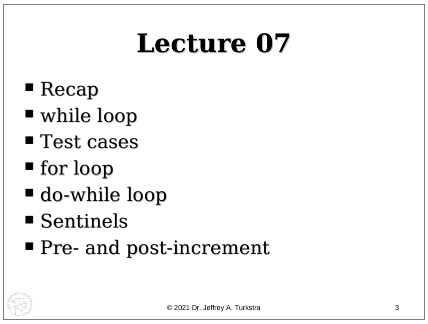
CS 180: Problem Solving and Object-Oriented Programming

Lecture 7: Iteration

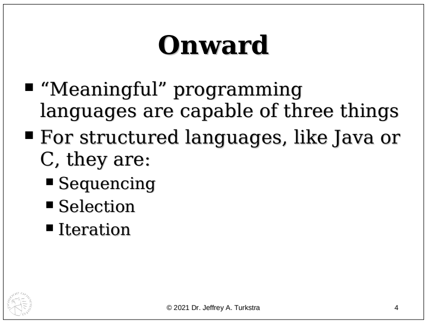
Dr. Jeff Turkstra


Reminder

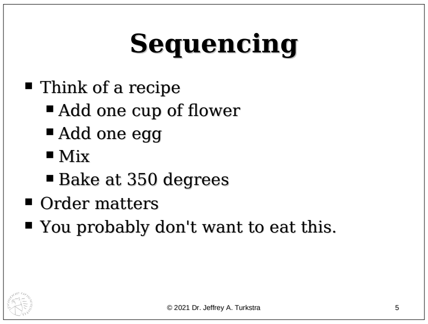
- We will have a regular lecture on Friday!


Lecture 07

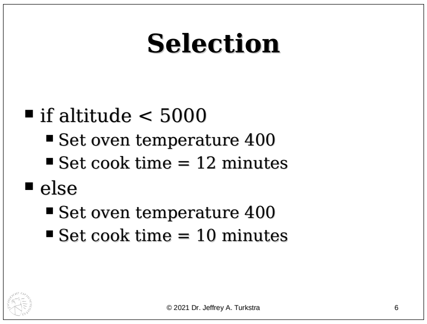
- Recap
- while loop
- Test cases
- for loop
- do-while loop
- Sentinels
- Pre- and post-increment


Onward

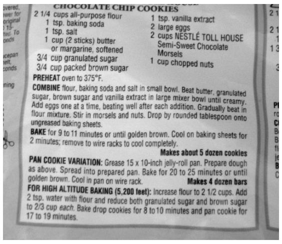
- “Meaningful” programming languages are capable of three things
- For structured languages, like Java or C, they are:
  - Sequencing
  - Selection
  - Iteration


Sequencing

- Think of a recipe
  - Add one cup of flower
  - Add one egg
  - Mix
  - Bake at 350 degrees
- Order matters
- You probably don't want to eat this.


Selection

- if altitude < 5000
  - Set oven temperature 400
  - Set cook time = 12 minutes
- else
  - Set oven temperature 400
  - Set cook time = 10 minutes



CHOCOLATE CHIP COOKIES

2 1/4 cups all-purpose flour
1 tsp. baking soda
1 tsp. salt
1 cup (2 sticks) butter or margarine, softened
3/4 cup granulated sugar
3/4 cup packed brown sugar
1 tsp. vanilla extract
2 large eggs
2 cups NESTLÉ TOLL HOUSE Semi-Sweet Chocolate Morsels
1 cup chopped nuts

PREHEAT oven to 375°F.

COMBINE flour, baking soda and salt in small bowl. Beat butter, granulated sugar, brown sugar and vanilla extract in large mixer bowl until creamy. Add eggs one at a time, beating well after each addition. Gradually beat in flour mixture. Stir in morsels and nuts. Drop by rounded tablespoon onto ungreased baking sheets.

BAKE for 9 to 11 minutes or until golden brown. Cool on baking sheets for 2 minutes; remove to wire racks to cool completely.

Makes about 5 dozen cookies

PAN COOKIE VARIATION: Grease 15 x 10-inch jelly-roll pan. Prepare dough as above. Spread into prepared pan. Bake for 20 to 25 minutes or until golden brown. Cool in pan on wire rack. **Makes 4 dozen bars**

FOR HIGH ALTITUDE BAKING (5,200 feet): Increase flour to 2 1/2 cups. Add 2 tsp. water with flour and reduce both granulated sugar and brown sugar to 2/3 cup *each*. Bake drop cookies for 8 to 10 minutes and pan cookie for 17 to 19 minutes.

Selection

- Alters a sequence
  - if...else if...else
  - switch
- And branching
  - break
  - continue
  - return

Iteration

- ...mix ingredients
- Place in pan, then oven
- Until golden brown,
  - Bake
- Remove from oven

Iteration

- Java provides three looping constructs

```
for (expr 1; expr 2; expr 3) {...}

while (expr) {...}

do {...} while (expr);
```

We are a strange loop

```
int i = 0;
while (i < 5) {
  System.out.println(“All work and no play”);
  i++;
}

i = 0;
do {
  System.out.println(“All work and no play”);
  i++;
} while (i < 5);

for (i = 0; i < 5; i++) {
  System.out.println(“All work and no play”);
}
```

Makes Jeff a dull boy

```
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
All work and no play
```

What happens when there is no advancement?

```
i = 0;
while (i < 5) {
  System.out.println(“Jeff is a dull boy”);
}

while (1 < 2) {
  System.out.println(“Jeff is a dull boy”);
}
```

Traffic light

- Demonstration

There must be a better way...

- Using loops for delays is (usually) a bad idea
  - Sometimes you have to, though. You'll know when.

```
Thread.sleep(unsigned int msec)
```

Iteration

- ...or repetition can be used to
  - Perform the same operation on different data
  - Accumulate information over a set of data
- Two parts
  - Block of code
  - Conditional determining when to halt

Two forms

- Indefinite: loop until done
  - No advance knowledge of number of iterations required
- Definite: loop a given number of times
  - Iterations controlled by a counter/size/limit

while

- Continually executes a block of statements while a condition is true
- Condition is a boolean expression

Problem: Odd or Even

- Write a program that reads integers from standard input and displays “odd” or “even” for each
- Stop when input no longer contains an integer
- `Scanner hasNextInt()`
  - True if another integer available
  - False otherwise, including EOF (ctrl-D)

```
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);
    }
  }
}
```

Problem: Palindrome

- Write a method in class Palindrome to test if the argument is a palindrome

  `Boolean isPalindrome(String s)`
- A palindrome is a string that reads the same backwards as forwards

```
import java.util.Scanner;

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;
  }
}
```

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"));
  }
}
```

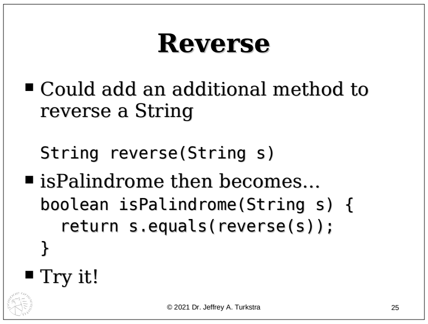
borg

$ javac -cp .:/usr/share/java/junit4.jar PalindromeTest.java

$ java -cp .:/usr/share/java/junit4.jar org.junit.runner.JUnitCore PalindromeTest

- Or…

  $ export CLASSPATH=$CLASSPATH:/usr/share/java/junit4.jar
- Make it permanent in .bashrc



Reverse

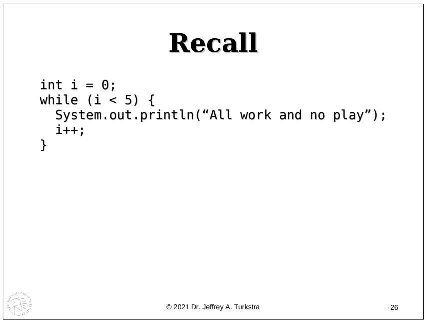
- Could add an additional method to reverse a String

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

- isPalindrome then becomes…

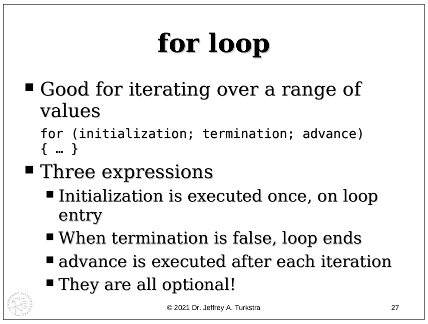
```
boolean isPalindrome(String s) {
  return s.equals(reverse(s));
}
```

- Try it!



Recall

```
int i = 0;
while (i < 5) {
  System.out.println("All work and no play");
  i++;
}
```

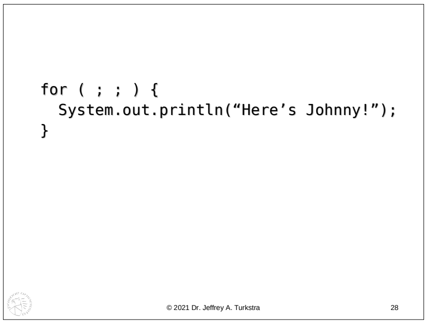


for loop

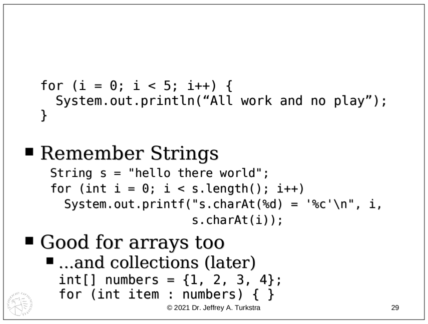
- Good for iterating over a range of values

```
for (initialization; termination; advance)
{ … }
```

- Three expressions
  - Initialization is executed once, on loop entry
  - When termination is false, loop ends
  - advance is executed after each iteration
  - They are all optional!



```
for ( ; ; ) {
  System.out.println("Here's Johnny!");
}
```



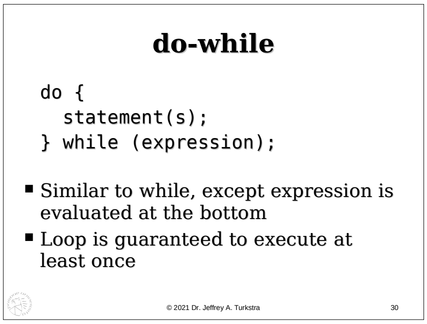
```
for (i = 0; i < 5; i++) {
  System.out.println("All work and no play");
}
```

- Remember Strings

```
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));
```

- Good for arrays too
  - ...and collections (later)

```
int[] numbers = {1, 2, 3, 4};
for (int item : numbers) { }
```



do-while

```
do {
  statement(s);
} while (expression);
```

- Similar to while, except expression is evaluated at the bottom
- Loop is guaranteed to execute at least once

Problem: Prompt for Even

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

```
import java.util.Scanner;

public class Prompter {
  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);
  }
}
```

Sentinel

```
import java.util.Scanner;

public class Sentinel {
  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 done = false;

    while (!done) {
      System.out.printf("Please enter an even number: ");
      n = in.nextInt();
      if (n % 2 == 0)
        done = true;
      else
        System.out.printf("The number %d is not even.\n", n);
    }

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

Elephants in Cairo

1. Go to Africa
2. Start at the Cape of Good Hope
3. Work northward in an orderly manner, traverse alternately east and west
4. During each pass
   - Catch each animal seen
   - Compare to a known elephant
   - Stop when a match is detected

- What happens if you don’t know how to swim?

No Drowning

1. Go to Africa
2. Put an elephant in Cairo
3. Start at the Cape of Good Hope
4. Work northward in an orderly manner, traverse alternately east and west
5. During each pass
   - Catch each animal seen
   - Compare to a known elephant
   - Stop when a match is detected

A note on incrementing

- `x++`, post-increment operator
  - Increments x by one, but expression value is the original x
- `++x`, pre-increment operator
  - Increments x by one, and expression value is the new x

Boiler Up!