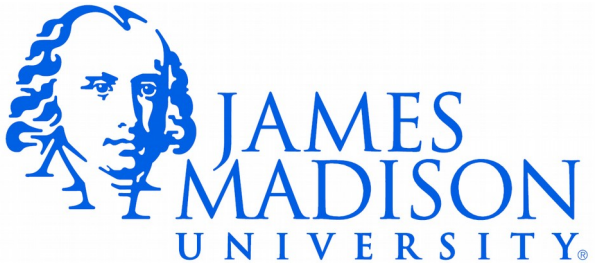
CS139 – Nested Loops



Loops and Scope

- Reminder – the scope of a variable is the part of the program where that variable is visible
- Will this compile?

```
while (number < 10) {
    String result = "latest " + number;
    number++;
}
System.out.println(result);
```

Loops and Scope

- Reminder – the scope of a variable is the part of the program where that variable is visible
- Will this compile?

```
while (number < 10) {
    String result = "latest " + number;
    number++;
}
System.out.println(result);
```

- No! `result` only exists in the block where it was declared

Loops and Scope

- Notice the difference between these loops

```
int i;
for (i = 0; i < 10; i++) {
   // Do some things.
}
```

```
for (int i = 0; i < 10; i++) {
   // Do some things.
}
```

Loops and Scope

- Notice the difference between these loops

```
int i;
for (i = 0; i < 10; i++) {
   // Do some things.
}
```

```
for (int i = 0; i < 10; i++) {
   // Do some things.
}
```

- What will be printed?

```
int i;
for (i = 0; i < 10; i++) {
   // Do some things.
}
System.out.println(i);
```

```
for (int i = 0; i < 10; i++) {
   // Do some things.
}
System.out.println(i);
```

Loops and Scope

- Notice the difference between these loops

```
int i;
for (i = 0; i < 10; i++) {
   // Do some things.
}
```

```
for (int i = 0; i < 10; i++) {
   // Do some things.
}
```

- What will be printed?

```
int i;
for (i = 0; i < 10; i++) {
   // Do some things.
}
System.out.println(i);
```

```
for (int i = 0; i < 10; i++) {
   // Do some things.
}
System.out.println(i);
```

Syntax error

Quiz #1

- What will be printed when this code executes?

```
for (int i = 0; i < 3; i++) {
    System.out.println(i);
}
```

Quiz #1

- What will be printed when this code executes?

```
for (int i = 0; i < 3; i++) {
    System.out.println(i);
}
```

```
0
1
2
```

Quiz #2

- What will be printed when this code executes?

```
for (int i = 0; i < 3; i++) {
    System.out.println(i);

    for (int j = 0; j < 2; j++) {
        System.out.printf(">> %d %d\n", i, j);
    }

}
```

Quiz #2

- What will be printed when this code executes?

```
for (int i = 0; i < 3; i++) {
    System.out.println(i);

    for (int j = 0; j < 2; j++) {
        System.out.printf(">> %d %d\n", i, j);
    }

}
```

```
0
>> 0 0
>> 0 1
1
>> 1 0
>> 1 1
2
>> 2 0
>> 2 1
```

- This is a nested loop.

Quiz #3

- What will be printed when this code executes?

```
for (int i = 0; i < 2; i++) {
    System.out.println(i);

    for (int j = 0; j < 2; j++) {
        System.out.printf("-- %d %d\n", i, j);
    }

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

}
```

Quiz #3

- What will be printed when this code executes?

```
for (int i = 0; i < 2; i++) {
    System.out.println(i);

    for (int j = 0; j < 2; j++) {
        System.out.printf("-- %d %d\n", i, j);
    }

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

}
```

```
0
-- 0 0
-- 0 1
** 0 0
** 0 1
** 0 2
1
-- 1 0
-- 1 1
** 1 0
** 1 1
** 1 2
```

Quiz #4

- What will be printed when this code executes?

  Be careful!

```
for (int i = 0; i < 3; i++) {
   System.out.println(i);

   for (int j = i; j < 2; j++) {
      System.out.printf("-- %d %d\n", i, j);
   }

}
```

Quiz #4

- What will be printed when this code executes?

  Be careful!

```
for (int i = 0; i < 3; i++) {
    System.out.println(i);

    for (int j = i; j < 2; j++) {
        System.out.printf("-- %d %d\n", i, j);
    }

}
```

```
0
-- 0 0
-- 0 1
1
-- 1 1
2
```

Naming Index Variables

- Why “i” and “j”? Aren't we supposed to pick meaningful names?
- Yes, but i and j are a widely used conventions for cases where:
  - We are only using the variable to keep track of how many times the loop has executed
  - We are using the variables to “i”ndex into some sequence...

Naming Index Variables

```
   public static int countX(String word) {
      int count = 0;

      for (int i = 0; i < word.length(); i++) {
         if (word.charAt(i) == 'X') {
            count++;
         }
      }

      return count;
}
```

Why Nested Loops

- Example of a nested loop in practice:

```
  public static void listPrimes(int max) {
     boolean noDivisors;

     for (int candidate = 2; candidate <= max; candidate++) {
        noDivisors = true;

        // Check the current candidate for primality
        for (int divisor = 2; divisor <= Math.sqrt(candidate); divisor++) {
           if (candidate % divisor == 0) {
              noDivisors = false;
           }
        }

        // If there were no divisors, it must be prime. Print it.
        if (noDivisors) {
           System.out.println(candidate);
        }
     }
}
```

Common Mistake

- See any problems?

```
for (int i = 0; i < 3; i++) {
   System.out.println(i);

   for (int j = 0; j < 2; i++) {
      System.out.printf(">> %d %d\n", i, j);
   }

}
```

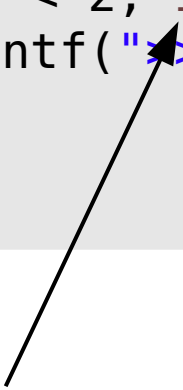
Common Mistake

- See any problems?

```
for (int i = 0; i < 3; i++) {
   System.out.println(i);

   for (int j = 0; j < 2; i++) {
      System.out.printf("=> %d %d\n", i, j);
   }

}
```



i is incremented here instead of j. Probably a copy-paste error.

Any Problems With This Method?

```
   public static boolean hasRepeatedCharacter(String word) {

      for (int i = 0; i < word.length(); i++) {
         for (int j = 0; j < word.length(); j++) {
            if (word.charAt(i) == word.charAt(j)) {
               return true;
            }
         }
      }

      return false;
}
```

Any Problems With This Method?

```
  public static boolean hasRepeatedCharacter(String word) {

     for (int i = 0; i < word.length(); i++) {
        for (int j = 0; j < word.length(); j++) {
           if (word.charAt(i) == word.charAt(j)) {
              return true;
           }
        }
     }

     return false;
}
```

- Two reasonable answers:
  - Multiple return statements violate the style guide. (OK with me in this case.)
  - Doesn't work correctly.

Fixed

```
   public static boolean hasRepeatedCharacter(String word) {

      for (int i = 0; i < word.length(); i++) {
         for (int j = i + 1; j < word.length(); j++) {
            if (word.charAt(i) == word.charAt(j)) {
               return true;
            }
         }
      }

      return false;
}
```

- Now it compares every letter to all of the subsequent letters.