CS 152
Computer Programming Fundamentals
Project 4 – Methods Practice Solution

Brooke Chenoweth

University of New Mexico

Fall 2026

maxOfThree

```java
/**
 * Returns largest of its arguments.
 * @param x First argument
 * @param y Second argument
 * @param z Third argument
 * @return Maximum of x, y and z
 */
public static int maxOfThree(int x, int y, int z) {
  if(x < y) {
    if(y < z) return z;
    else return y;
  } else {
    if(x < z) return z;
    else return x;
  }
}
```

maxOfThree – local variable

```
/**
 * Returns largest of its arguments.
 * @param x First argument
 * @param y Second argument
 * @param z Third argument
 * @return Maximum of x, y and z
 */
public static int maxOfThree(int x, int y, int z) {
   int largest = x;
   if(y > largest) {
      largest = y;
   }
   if(z > largest) {
      largest = z;
   }
   return largest;
}
```

maxOfThree – ternary operator

```
/**
 * Returns largest of its arguments.
 * @param x First argument
 * @param y Second argument
 * @param z Third argument
 * @return Maximum of x, y and z
 */
public static int maxOfThree(int x, int y, int z) {

    return
        x < y ? (y < z ? z : y) : (x < z ? z : x);

     // like an if statement, but is an expression
     // BooleanExpr ? ValueIfTrue : ValueIfFalse
}
```

maxOfThree – sorting

```
/**
 * Returns largest of its arguments.
 * @param x First argument
 * @param y Second argument
 * @param z Third argument
 * @return Maximum of x, y and z
 */
public static int maxOfThree(int x, int y, int z) {
  // This is really overkill for just 3 values
  int[] nums = {x, y, z};
  java.util.Arrays.sort(nums);
  return nums[2];
}
```

isVowel

```
/**
 * Is the character given a vowel?
 * A vowel is one of a, e, i, o, or u (upper or lower
 * @param c Character to check
 * @return True if c is a vowel, false if not
 */
public static boolean isVowel(char c) {
  return c == 'a' || c == 'e' || c == 'i'
      || c == 'o' || c == 'u'
      || c == 'A' || c == 'E' || c == 'I'
      || c == 'O' || c == 'U';
}
```

isVowel – switch, toLowerCase

```
/**
 * Is the character given a vowel?
 * A vowel is one of a, e, i, o, or u (upper or lower
 * @param c Character to check
 * @return True if c is a vowel, false if not
 */
public static boolean isVowel(char c) {
    switch(Character.toLowerCase(c)) {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
        return true;
    default:
        return false;
    }
}
```

isVowel – String method

```
/**
 * Is the character given a vowel?
 * A vowel is one of a, e, i, o, or u (upper or lower
 * @param c Character to check
 * @return True if c is a vowel, false if not
 */
public static boolean isVowel(char c) {
  return "aeiouAEIOU".indexOf(c) >= 0;
}
```

Returning boolean expressions

If you have code of the form

```
if(booleanExpr) {
  return true;
} else {
  return false;
}
```

you can write it more concisely as

```
    return booleanExpr;
```

indexOfLastVowel – loop backwards

```java
/**
 * Where is the location of the last vowel in the giv
 * @param x String to check
 * @return 0 based location of last occurance of a vo
 *          -1 if there are no vowels present.
 */
public static int indexOfLastVowel(String x) {
    for(int i = x.length() - 1; i >= 0; i--) {
        if(isVowel(x.charAt(i))) {
            return i;
        }
    }
    return -1;
}
```

indexOfLastVowel – variable, forwards

```
/**
 * Where is the location of the last vowel in the giv
 * @param x String to check
 * @return 0 based location of last occurance of a vo
 *          -1 if there are no vowels present.
 */
public static int indexOfLastVowel(String x) {
    int index = -1;
    for(int i = 0; i < x.length(); i++) {
        if(isVowel(x.charAt(i))) {
            index = i;
        }
    }
    return index;
}
```

toggleCase – char math

```
/**
 * Flip character's case between upper and lower.
 * If lowercase, return uppercase equivalent.
 * If uppercase, return lowercase equivalent.
 * If not a letter, return unchanged.
 * @param c Character to process
 * @return Character with case switched.
 */
public static char toggleCase(char c) {
  int offset = 'A' - 'a';
  if('a' <= c && c <= 'z') {
    return (char)(c + offset);
  } else if('A' <= c && c <= 'Z') {
    return (char)(c - offset);
  } else {
    return c;
  }
}
```

toggleCase – Character methods

```java
/**
 * Flip character's case between upper and lower.
 * If lowercase, return uppercase equivalent.
 * If uppercase, return lowercase equivalent.
 * If not a letter, return unchanged.
 * @param c Character to process
 * @return Character with case switched.
 */
public static char toggleCase(char c) {
  if(Character.isLowerCase(c)) {
    return Character.toUpperCase(c);
  } else if(Character.isUpperCase(c)) {
    return Character.toLowerCase(c);
  } else {
    return c;
  }
}
```

toggleCase

```java
/**
 * Return a string with each lowercase letter
 * converted to uppercase, each uppercase letter
 * switched to lowercase, and each non-letter
 * character left unchanged
 * @param s The string to process
 * @return New string with upper and lower case switc
 */
public static String toggleCase(String s) {
    String result = "";
    for(int i = 0; i < s.length(); i++) {
        char c = s.charAt(i);
        result += toggleCase(c);
    }
    return result;
}
```

isVowelless – helper for averageVowelless

```
/**
 * Is the given string vowelless?
 * @param s String to check
 * @return True if s contains no vowels,
 *         false if it has a vowel
 */
public static boolean isVowelless(String s) {
    return indexOfLastVowel(s) < 0;
}
```

We could just do this test directly in the averageVowelless method, but the method call fits better on the next slide.

averageVowelless – the long way

```
public static double averageVowelless(
                String a, String b, String c,
                String d, String e) {
  int count = 0;
  double sum = 0.0;

  if(isVowelless(a)) {
    count++;
    sum += a.length();
  }
  if(isVowelless(b)) {
    count++;
    sum += b.length();
  }
  if(isVowelless(c)) {
    count++;
    sum += c.length();
  }
  if(isVowelless(d)) {
    count++;
    sum += d.length();
  }
  if(isVowelless(e)) {
    count++;
    sum += e.length();
  }
  if(count == 0) {
    return -1111;
  } else {
    return sum/count;
  }
}
```

averageVowelless – array

```
public static double averageVowelless(
                String a, String b, String c,
                String d, String e) {
    int count = 0;
    double sum = 0.0;

    String[] strs = new String[]{a, b, c, d, e};

    for(int i = 0; i < strs.length; i++) {
        if( isVowelless(strs[i]) ) {
            count++;
            sum += strs[i].length();
        }
    }

    if(count == 0) return -1111;
    else return sum/count;
}
```

averageVowelless – foreach loop

```
public static double averageVowelless(
                String a, String b, String c,
                String d, String e) {
    int count = 0;
    double sum = 0.0;

    String[] strs = new String[]{a, b, c, d, e};

    for(String s : strs) {
        if( isVowelless(s) ) {
            count++;
            sum += s.length();
        }
    }

    if(count == 0) return -1111;
    else return sum/count;
}
```

getTotalBill

```
/**
 * Calculates meal total after adding a tip.
 * @param meal Cost of meal. Must be positive
 * @param tip Tip percentage
 *            (must be between 0 and .95)
 * @return Total bill amount after adding tip.
 */
public static double getTotalBill(int meal,
                                  double tip) {

  if(meal <= 0 || tip < 0 || tip > 0.95) return -1;

  return meal * (1 + tip);
}
```

getTotalBill – another version

```
/**
 * Calculates meal total after adding a tip.
 * @param meal Cost of meal. Must be positive
 * @param tip Tip percentage
 *            (must be between 0 and .95)
 * @return Total bill amount after adding tip.
 */
public static double getTotalBill(int meal,
                                  double tip) {

  if(meal > 0 && tip >= 0 && tip <= 0.95) {
    double tipVal = meal*tip;
    double total = meal + tipVal;
    return total;
  } else {
    return -1;
  }
}
```

solveQuadratic

```java
/**
 * Find larger of the solutions of quadratic equation
 * Assumes tests only with real solutions (no complex roots! no div by
 * @param a Quadratic coefficient
 * @param b Linear coefficient
 * @param c Constant coefficient
 * @return Larger of the two solutions
 */
public static double solveQuadratic(double a, double b, double c) {
    // for a x^2 + b x + c = 0
    // roots are x = (-b +- sqrt(b^2 - 4ac))/(2a)

    double discrim = b * b - 4 * a * c;
    double sqrtDisc = Math.sqrt(discrim);
    double num1 = -b - sqrtDisc;
    double num2 = -b + sqrtDisc;
    double denom = 2*a;

    double root1 = num1 / denom;
    double root2 = num2 / denom;

    return Math.max(root1, root2);
}
```