Types, Naming, and Table-Driven Methods
Feedback on code review sessions?
Any questions about the reading?
Biggest Takeaways from Ch. 11

- names should be explanatory
- good names
  - describe problems (not solutions)
  - tell what (not how)
- ideally 8-20 characters (or better yet 10-16)
- use longer names for rarely used variables, variables with longer lifetimes
- okay to use i,j,k for tight loops, shouldn't leave loop scope
- give booleans/flags meaningful names
- suggests using 'is' in boolean names (e.g., isWorking, isReady)
- temp is not a variable name
- name constants for their meaning not their value (e.g., CYCLES_NEEDED, not FIVE)
Naming Conventions

- types vs variables
  - in Java: ThisIsAnotherType thisIsAnObject
- globals: g_something
- member: m_something_else
- constants: PI
Scrabble
Scrabble word score

- Sum of the letter values

English-language editions of Scrabble contain 100 letter tiles, in the following distribution:

- 2 blank tiles (scoring 0 points)
- 1 point: E $\times$12, A $\times$9, I $\times$9, O $\times$8, N $\times$6, R $\times$6, T $\times$6, L $\times$4, S $\times$4, U $\times$4.
- 2 points: D $\times$4, G $\times$3.
- 3 points: B $\times$2, C $\times$2, M $\times$2, P $\times$2.
- 4 points: F $\times$2, H $\times$2, V $\times$2, W $\times$2, Y $\times$2.
- 5 points: K $\times$1.
Scrabble word score, continued

public static int wordScore(String word) {
    int score = 0;
    for (int i = 0 ; i < word.length() ; i++) {
        char letter = word.charAt(i);
        score += letterScore(letter);
    }
    return score;
}
public static int letterScore(char c) {
    char upperC = Character.toUpperCase(c);
    switch (upperC) {
    case 'A':
    case 'E':
    case 'I':
    case 'L':
    case 'N':
    case 'O':
    case 'R':
    case 'S':
    case 'T':
    case 'U':
        return 1;
    case 'D':
    case 'G':
        return 2;
    case 'B':
    case 'C':
    case 'M':
    case 'P':
    case 'F':
    case 'H':
    case 'V':
    case 'W':
    case 'Y':
        return 4;
    case 'K':
        return 5;
    case 'J':
    case 'X':
        return 8;
    case 'Q':
    case 'Z':
        return 10;
    default:
        // handle error
    } // should never reach here
    return 0;
}
Table-based Solution

```java
private static final int[] scoresByChar =
    {/* A */ 1, /* B */ 3, /* C */ 3, /* D */ 2, /* E */ 1,
    /* F */ 4, /* G */ 2, /* H */ 4, /* I */ 1, /* J */ 8,
    /* K */ 5, /* L */ 1, /* M */ 3, /* N */ 1, /* O */ 1,
    /* P */ 3, /* Q */ 10, /* R */ 1, /* S */ 1, /* T */ 1,
    /* U */ 1, /* V */ 4, /* W */ 4, /* X */ 8, /* Y */ 4,
    /* Z */ 10};

public static int letterScore2(char c) {
    char cAsUppercase = Character.toUpperCase(c);
    int index = cAsUppercase - 'A';
    if (index < 0 || index >= 26) {
        // handle error
    }
    return scoresByChar[index];
}
```

"""A```B```B```C```C```C```D```O```E```F```F```F"""
Map < Character, Integer > ScrabbleValueMap =
new HashMap < Character, Integer >(
    ScrabbleValueMap.put ("A", SCRABBLE_VALUE_FOR_A),
    ScrabbleValueMap.get (c As Upper-case),
);
To Dos for Thursday

- Read Ch. 14 (Organizing Straight-Line Code)
- Read Ch. 15 (Using Conditionals)
- Read Ch. 16 (Controlling Loops)
- Read Ch. 17.1 and 17.2 (Multiple returns & Recursion)
  - 17.3+ is optional given that it doesn’t really relate to Java

- Write code to:
  - Parse a JSON description of a series of rooms
  - Verify that if you can get from A to B, that you can get from B to A.
JSON format

- **Direction:**
  - direction: String
  - room: String  // a Room’s name

- **Room:**
  - name: String
  - description: String
  - directions: Direction []

- **Layout:**
  - initialRoom: String  // a Room’s name
  - rooms: Room []
{  
  "initialRoom": "MatthewsStreet",
  "rooms": [
    {
      "name": "MatthewsStreet",
      "description": "You are on Matthews, outside the Siebel Center",
      "directions": [
        {
          "direction": "East",
          "room": "SiebelEntry"
        }
      ]
    },
    {
      "name": "SiebelEntry",
      "description": "You are in the west entry of Siebel Center. You can see the elevator, the ACM office, and hallways to the north and east.",
      "directions": [
        {
          "direction": "West",
          "room": "MatthewsStreet"
        },
        {
          "direction": "Northeast",
          "room": "AcmOffice"
        },
        {
          "direction": "North",
          "room": "SiebelNorthHallway"
        }
      ]
    }
  ]
}