# Lab #3 – Decisions

Total Possible Points: 5

DUE: Friday, October 19th @ 11:59PM

LATE POLICY: -1 point off per day; no submissions accepted after Wed. Oct 24 @ 11:59PM

## What to Submit

* An Eclipse project in a ZIP folder named **Lab3.zip** or a RAR archive named **Lab3.rar**.

## Goals

* To become more familiar with the Scanner class and its methods
* To write programs that make decisions
* To write Boolean expressions that correctly compare two String objects
* To better understand Java if, if-else, and if-else if block syntax
* To write programs that accept AND VALIDATE user input
* To use nested decision blocks and Boolean operators to make multi-faceted decisions

## Introduction

**In this lab, you will implement a program that simulates a coin flip.**

The infamous 1998 Thanksgiving Day "coin flip fiasco" in which referee Phil Luckett (center) is accused of mishearing the call of Pittsburgh Steelers running back Jerome Bettis (left). The host Detroit Lions won the toss and the game, 19-16 in OT. Enhanced sound analysis of the broadcast confirmed that Mr. Bettis, in fact, called both "heads" and "tails" while the coin was in air. You're not supposed to do that. (Photo: steelersgab.com)

The computer "flips" a coin using the Random number generator class. The user calls "heads" or "tails", and the program determines if the user won or lost. It also validates the user's input – if they type anything other than "h" (heads) or "t" (tails), the program alerts them.

**NOTE**: we have not covered loops yet, so much like Exercise 6's game, this program only performs a single coin flip.

|  |  |
| --- | --- |
| Warning | **Where to Get Help**  Scanner *class*   * Chapter 4 Lecture Slides, 27-32 * Exercises 3, 4, and 6   *Decisions & Comparisons*   * Chapter 5 Lecture Slides, 3-38 * Exercise 6 * Basic code sample: ch05.section\_01 > ElevatorSimulation.java * Comparisons code sample: ch05.section\_02 > Compare.java * Multiple alternatives code sample: ch05.section\_03 > Earthquake.java * Nested decisions code sample: ch05.section\_04 > TaxReturn.java   *Comparing Strings*   * Chapter 5 Lecture Slides, 22 * Code Sample: ch05.section\_02 > Compare.java   *Boolean Operators*   * Chapter 5 Lecture Slides, 44-54 * Code Sample: ch05.section\_07 > Compares.java |

## Part 1: Setting Up Your Eclipse Project (0 points)

* Create a new Java Project named **Lab 3**.
* Create a package named game
* Create a new Java class named CoinFlipper inside the game package. Add a public static void main(String[] args) method to the class.

**All code goes inside the main() method.**

## Part 2: Adding a Random Number Generator (0.5 point)

As in Exercise 6, we'll use the Random number generator class to choose a random number. Here, Random will act as a coin flipper. We call its nextInt() method to choose from just two numbers: 0 (heads) and 1 (tails).

**YOUR TURN**: Add the following statements to the beginning of your main() program, as well as the appropriate import statement above the class declaration statement:

// flip a coin

Random r = **new** Random();

**int** coin = r.nextInt(2);

|  |  |
| --- | --- |
| Lightbulb | If Eclipse doesn't add the import statement for you automatically, you'll see *red jagged lines* under the words Random. Java's compiler and Eclipse are trying to tell you something: *they don't know what a "Random" is*. *You have a problem*.  Fix it by putting your mouse cursor over the word "Random", holding your cursor there for a few seconds. A tooltip appears with possible solutions (sorcery!) Choose the first suggestion to add the import statement. |

## Part 3: Adding Constants (0.5 point)

**YOUR TURN**: **Add two (2) constant integer variables** to store the values of the two possible outcomes: *0* for "heads" and *1* for "tails". Use clear, appropriate names that follow Java's conventions for constants.

## Part 4: Getting the User's Call (0.5 point)

**YOUR TURN**: **Add code that asks the user to choose heads or tails.**

* The user MUST enter either "h" for heads or "t" for tails. You may NOT tell them to enter 0, 1, or anything else but those two choices. Make the prompt clear!
* Get the user's input and store it in a new String variable with a relevant name.

## Part 5: Validating the User's Input (1 point)

**FIRST RULE OF USER INTERFACES: NEVER TRUST THE USER'S INPUT!** Perhaps it was just an innocent mistake – or something more – but bad input breaks programs. Compromised programs lead to serious financial, economic, or reputational damage – or even worse. The benefits of validation code far outweigh the costs of adding lines of additional logic.

Right, the coin flipping program. We need to verify that the user entered something valid: either "h" for heads or "t" for tails. If they didn't, we 1.) alert them and 2.) bypass the rest of the game logic. If they didn't pick heads or tails, how can we play the game?

**YOUR TURN**: Using correct logic and Java syntax, write a decision block that **tests the user's input according to the following pseudocode**:

If the user did not enter "h" AND the user did not enter "t"

Print "Sorry, that was a bad input."

Else

// Leave empty for now; will add code in Parts 6 & 7

**Save your work and run your program. Type in something other than "h" or "t". What happens?**

## Part 6: Using Nested Decisions and Building a String (1 point)

Very often, we write logic in programs that print custom output based on a user's decision or some set of outcomes. For example, our coin flipping program must print the two (2) possible results of the coin flip

The coin came up heads

or…

The coin came up tails

You can write the correct logic in a few ways. **As practice for the rest of the semester**, your instructor has one specific way in mind: *by building up a String object in pieces, then printing the result*.

**YOUR TURN**: Inside the empty Else block from Part 5, write the Java code that **prints the outcome of the coin flip according to the following pseudocode**:

Set String result = "The coin came up "

If the coin came up heads

Append "heads" to result

Else

Append "tails" to result

Print result

|  |  |
| --- | --- |
| Lightbulb | **HINT**: Reuse variables and constants from earlier portions of the program! |

## Part 7: Using Multiple Alternatives and Boolean Operators (1.5 point)

We're down to the last (and perhaps most complicated) part of the program. There are three (3) possible outcomes to the game:

1. The player picks heads AND the coin ends up "heads" => **player wins**
2. The player picks tails AND the coin ends up "tails" => **player wins**
3. Otherwise => **player loses**

**YOUR TURN**: Immediately after the code you just wrote for Part 6, write the Java code that **prints whether the player wins or loses based on the rules above**. Your code must:

* Use a multiple alternative decision block
* Use the correct Boolean operator to test two conditional expressions in the same statement.

**Save your work and run your program. Play it several times to see if the game behaves at you expect. When you're satisfied with your work, ZIP or RAR your entire Lab 3 project and upload to Moodle.**