# Basketball

To receive full credit, you must follow the steps and record reflections as you make progress.

This assignment assumes that you already completed the **Basketball Warmup** assignment.
**A reminder of the requirements, from that document:**
A customer would like to run a simulation of basketball players that compete in shooting contests. Each basketball player has a name and an accuracy which reflects the chance that he or she will make a given shot. A shooting contest will have a title and a certain number of shots that are included. During a shooting contest, each player takes all of their shots before the next player then has a turn to take all of their shots. The customer would like to be able to add multiple players to a contest, decide when the contest is run, and be able to view a report about a given contest for a specific player. One piece of information to report for a contest is the total percentage of shots made. Another piece of information to report is the maximum number of shots made in a row. Finally, there should be a way to view the hits and misses (via textual representation) of a player like this:
__X__X_XX_XXXX_X___X
Where an "X" indicates a made shot and "_" indicates a miss. In the above example, the maximum hit streak was 4 because four shots were made in a row (XXXX).

**Part A: UML diagram of your original code**
Create a UML diagram of your design for the Warm Up Basketball specification as you had coded it. This should be created using a computer generated graphics (NOT hand-written) – use UMLet (easy drag and drop) or PlantUML (sort of a coding language which is what we use to generate diagrams for class). PlantUML has a Google Doc chrome extension which might be nice to use if you need to collaborate remotely.

**To submit it**, (1) checkout the Basketball project from SVN. (2) Save a pdf of your UML diagram in the project. (3) Commit both the UML source file (.uxf if UMLet) and the pdf (be sure to select the Basketball project before choosing "commit" and check to see that it sees the new file). If you used PlantUML, just save and commit a text file with the link to the google doc of your UML.

**Part B: New and improved UML diagram**
Create a second UML diagram of a design that would solve this problem. **Make this design follow the principles as per the Object-oriented Design Principles sheet you received in class. <u>This second version will need to work with a console-based UI that we give you. Make sure to look at, from what you checked out from SVN, the starter files and methods requiring implementation before writing making this new UML diagram!</u>**
*Once your new UML design has been created, FREEZE this first version for submission as your first draft.*

To submit it, follow the same instructions as above.

**Part C: Implement Your New Design**
Using your newly UML design, create code that implements the design of the UML. It is quite possible that you had a problem in your original UML (this is OK!). If you discover you have to make changes to your design, then create a new UML diagram and write a summary on what you had to change and why (see reflection question #2 in part D). Our goal here is to assess if you can successfully translate UML-to-code, code-to-UML, and also understand on a more concrete level why design principles can make your code easier to use.

**Part D: Reflection**
Write in responses to the questions provided in the reflection_questions.txt file located in your Basketball repository.

**Starter Files:**
You are given two starting files and a test file:

*BasketballUtility* - this gives you a function to generate random variables in a repeatable way (like in the warm-up)

*BasketballMain* - this allows you to run commands from the console

You are asked to implement the follow methods:

```
public void handleCreatePlayer(String name, double accuracy) {
public void handleCreateContest(String contestName, int shots ) {
public void handleAddPlayerToContest(String playerName, String contestName) {
public void handleRunContest(String contestName) {
public String handleGetRawData(String playerName, String contestName ) {
public int handleGetHitStreak(String playerName, String contestName ) {
public double handleGetPercentage(String playerName, String contestName ) {
```

You are also given sample text to enter in your complete code to test it out. These are located in file(s) labeled input*.txt in the main Basketball folder.

Here is sample output of using a completed version of the program. You can imitate this using the commands in yellow (also provided in input1.txt ).

```
Welcome to BasketBall.  Enter commands.  Type 'exit' to end.
create-player Abby 0.7
Log: player "Abby" created
create-player Bob 0.6
Log: player "Bob" created
create-player Chad 0.3
Log: player "Chad" created
create-contest Grandtoss 20
Log: contest "Grandtoss" created
add-player-to-contest Abby Grandtoss
Log: player "Abby" added to contest "Grandtoss"
add-player-to-contest Bob Grandtoss
Log: player "Bob" added to contest "Grandtoss"
add-player-to-contest Chad Grandtoss
Log: player "Chad" added to contest "Grandtoss"
run-contest Grandtoss
Log: Contest "Grandtoss" was run
view-contest-report Abby Grandtoss
Reporting Stats for Player: Abby in Contest: Grandtoss
Raw Data:   _X_XXXXXX_X_XXX____X
Hit Streak: 6
Percentage: 60.0%
Log: Viewing report on Abby in contest: Grandtoss
view-contest-report Bob Grandtoss
Reporting Stats for Player: Bob in Contest: Grandtoss
Raw Data:   XX_X_XXX_XX_XXXXXXX_
Hit Streak: 7
Percentage: 75.0%
Log: Viewing report on Bob in contest: Grandtoss
view-contest-report Chad Grandtoss
Reporting Stats for Player: Chad in Contest: Grandtoss
Raw Data:   _X__X___X_X_X_______
Hit Streak: 1
Percentage: 25.0%
Log: Viewing report on Chad in contest: Grandtoss
```