**CS203 HW 7**

**GamePlayer Class Implementation**

**Spring 2012**

**Due Date:** Wednesday, Mar 21 at 11 AM (Wednesday after spring break)

You are creating a GamePlayer class for a new game. The game designers ask that you implement a Java class called GamePlayer with the following specification. The class represents players in a car racing game where the drivers try to collect coins. The idea is that the player collects coins, the size of the car can grow and shrink, and the car itself can flash.

**Specification for GamePlayer class:**

Attributes (instance variables): name, isFlashing, size, coins, catchPhrase [**done for you**]

Actions (methods):

1. A constructor – the constructor should have parameters for name and catchphrase. The size (of the car) should be assigned to 60.0 (60 inches) and the coins should be assigned to 0. The isFlashing instance variable should be set to false. [**done for you**]
2. getName – returns name of player (as a String) [**done for you**]
3. (.25 points) getSize – returns size of car (as a double)
4. (.25 points) getCoins – returns the number of coins the player has (as an int)
5. (.5 points) setToFlash – assigns isFlashing to true
6. (.5 points) earnCoins – increases coins by amount (passed as an int parameter). If amount is negative, the method should return false and not update the coins.
7. (.5 points) loseCoins – decreases coins by amount (passed as an int parameter). If the amount is negative, the method should return false. If the amount is bigger than the player’s current coins, the coins should be set to 0 and the method should return true [A player cannot lose more coins than it has]. Otherwise, coins should be reduced by amount and the method should return true.
8. (.5 points) growCarBy – increases the size of the car (passed as a double). If the number of inches is negative, the size should shrink by the inches passed in. The minimum size of a car is 40 inches and the maximum size is 80 inches. When the size is at one of these extremes, it cannot grow or shrink beyond these numbers. For example, if the size is 60 inches and grows by 25.6 inches, it can only grow to 80 inches.
9. (.5 points) holler – prints the player’s name followed by their catchPhrase. For example, this method might print “Joe says: What’s up?”.
10. (1 point) takeCoinsFrom – should take a GamePlayer as a parameter and steal all of its coins (updating both player’s point totals correctly)
11. (1 point) display – prints the player’s characteristics. For example, this method might print “Joe’s car is flashing and is 75.0 inches in size. Joe has 32 coins.”

The game designers are giving you some latitude to help with creating the game. They ask that you implement (1 point):

1. One additional *instance variable* for a GamePlayer object. You may need to update the constructor for this requirement.
2. One additional *method* using the new instance variable.

**Specification for Main class:**

(1 pt) After or while implementing your GamePlayer class, you should design a Main class that contains the main method to test your GamePlayer class. Your main method should test all methods in the GamePlayer class. [Note: you will need to create at least two players to ensure the takeCoinsFrom method works properly.]

UML Description of GamePlayer

GamePlayer

- name: String

- isFlashing: boolean

- size: double

- coins: int

- catchphrase: String

- (your choice)

----------------------------------------------------------------------------------------------------

+ GamePlayer(String playerName, String playerPhrase) [constructor]

+ getName(): String

+ getSize(): double

+ getCoins(): int

+ setToFlash(): void

+ earnCoins(int amount): boolean

+ loseCoins(int amount): boolean

+ growCarBy(double inches): void

+ holler(): void

+ takeCoinsFrom(GamePlayer victim): void

+ display(): void

+ (your choice)

**Additional Enrichment**

1. Create a Game class that creates GamePlayers and has an actual game goal.
2. Create graphics for your players and their cars [more difficult]

**Logistics**

1. Download starter project from Moodle [You may start your own BlueJ project if you want to start from scratch]
2. Be sure to use appropriate Java comments in your code, so that the javadoc can be automatically created for your GamePlayer class.
3. Turn in your BlueJ project (all files) and summary report as a single .zip file. Put your summary report in the folder containing your BlueJ project. Right-click on the folder containing all files and find the menu option for (SendTo -> Compressed). Name the file username\_HW7.zip.

**Grading Guidelines**

Your program will be graded on a scale of 0 to 7 in two categories:

* Code Quality: Design, Specification, Implementation, Documentation
* Code Functionality: Functionality of code and adherence to project specification

Your summary report will be graded on a scale of 0 to 6 based on:

* Correct use of technical vocabulary
* Evidence of testing
* Clarity
* Organization / Headings

**Report Guidelines and Format: (NOTE: These are questions specific to this assignment.)

1. Introduction and System Use:** Instead of writing text for this part of the written summary, **create Java documentation** for your GamePlayer class. You can create the documentation by going to Tools->(Project Documentation) in BlueJ. As long as you used the comment tags (@author, @version, @param, @return) in your code, the documentation should all be generated automatically and correctly for you. Copy and paste your documentation into your written report or include it as a separate file in your submission.

**2. System Description:**

**2.1:** **Method description**: Copy the code for the growCarBy method here. Describe what happens when the amount passed in is negative. Describe what happens when the amount passed in is >= 0.

**2.2: Class description:** Describe what your new instance variable and new method model in the GamePlayer class.

**3. Testing and Evaluation:**

* 1. **Adherence to Specification:** If your code adheres to the specification, state “Adheres to specification”. If there are additions or parts that are not working, describe them here.
	2. **Completeness**: Argue that you have tested **\*all\*** your GamePlayer methods and typical/edge/invalid inputs as parameters to all methods as evidenced by the code in your main method. You probably want to print out information about the GamePlayer objects that you create. For example, if you have created bobPlayer, you can test the earn method as follows:
	boolean earnOK = bobPlayer.earnCoins(100);
	System.out.println("Bob's earn was ok: " + earnOK);
	System.out.println("Bob's current coins: " + bobPlayer.getCoins());
	3. **Sample output**: Include a printout of the output produced by running your main method.

**4. Conclusion:** Describe the most challenging aspect(s) of this program and what you learned by completing it.

1: How long did you spend on this assignment?

2: By typing your name here, you are acknowledging that the code and report you are submitting are your own. If you received help from people other than Tammy, include that information here.

**Appendix:** Copy and paste your code here (use Courier font so the characters line up correctly).