University of Nebraska – Kearney
CSIS 150: Object Oriented Programming
Fall 2017
HW1 – Classes & Arrays

Due: 9/22//2017
100 points possible

Objectives

1. Use IntelliJ development environment for implementing object-oriented problem solving techniques.
2. Work with basic Java tools (methods, loops, decisions, assignment statements).
3. Practice defining classes (including fields and methods).
4. Practice using arrays.

Requirements

1. The homework is expected to be submitted on time. Late work will be accepted; however you lose 10% of your points for each late day.
2. You can get help from the instructor, the tutor, or other students. However, the homework must be finished by yourself, and you should indicate it in the program heading if you got any help.
3. **The first 13 questions should be answered in the heading documentation of your program.**
4. The programming part should be completed in Java and submitted through Blackboard.

Multiple Choice: Questions 1-10 are multiple-choice questions. Each one has one correct answer.

1. [2 points] The keyword that is used to create an object variable in memory is:
   A. public B. int C. new D. Class

2. [2 points] An object is a(n)
   A. blueprint B. instance of some class C. variable D. cookie cutter

3. [2 points] This is a method that return the value of a class's field
   A. accessor B. constructor C. void D. mutator

4. [2 points] When the value of an item is dependent on other data, and that item is not updated when the other data is changed, what has the value become?
   A. bitter B. moldy C. asynchronous D. stale

5. [2 points] To prevent other classes from accessing a class's field (i.e., data member) at all, which of the following access specifier should be used?
   A. private B. public C. protected D. Do not use any access specifier

6. [2 points] If you provide any constructor for a class, Java compiler will no longer provide the default no-arg constructor. Is this statement true or false?
   A. True B. False

7. [2 points] Read the following piece of code

```
public class Demo {
public void demo (String param) {
    ...;
}
public int demo (int param) {
    ...;
}
}
```

The demo methods in the class Demo are an example of overloading. Is this statement correct?
A. True B. False

8. [2 points] A static method cannot access instance fields of a class. Is this statement true or false?
A. true B. false

9. [2 points] We have class Demo defined as follows:

```
public class Demo {
    public static int counter = 0;
    public int increase() {
         return counter++;
    }
}
```

Now we have the following two statements in the main method that use the increase method as follows:

```
System.out.print(Demo.increase());
System.out.println(Demo.increase());
```

The output will be:
A. 0 0 B. 1 1 C. 1 2 D. 0 1 E. error

10. [2 points] Each object (instance) of a class has its own set of fields (member variables).
A. true B. false

11. [2 points] When an object is passed as an argument to a method, this is actually passed.
A. a copy of the object
B. a reference to the object
C. the name of the object
D. none of these; you cannot pass an object

12. [2 points] Assume that h1 is an object that reference the House class, as defined in our CSIT 150 class, the statement to create the h1 object is:

A. `h1 = new House();`
B. `House h1 = new House;`
C. `House h1 = new House();`
D. `new House = h1;`
E. `none of the above.`

13. [6 points] Find and fix the error(s) **and write two appropriate accessor methods and two appropriate constructors.**

```
public class MyClass {
private int x;
private double y;
public static void setValues(int a, double b) {
      x=a;
      y=b;
}
```

Programming Problem: TicTacToe

You are implementing a software game that plays TicTacToe. To help you build the game system, the detailed design is provided as follows:

1. Class `Player`

   This class is used to describe the player and actions on a player. To be more specific, class `Player` should have at least the following two fields (i.e., data members):

   - `char name`, which is the name of the player (X or O)
   - `int gamesWonCount`, which is the number of games that this player has won
   - other fields that you want to add.

You also need to provide the following methods:

- `Player(char inName)`, which is the constructor to instantiate this class
- `Appropriately named getters and setters that you feel are needed (or desired)`
- `toString()`, which returns the information about this player.
- Any other methods that you feel that you need to use.

2. Class `GameBoard`

   This class is used to manage the board for a TicTacToe Game. To be more specific, class GameBoard should have:

   - a 3X3 array that stores the current game board value.
   - Other fields that you want to add.

   You also need to provide the following methods:

   - `Default constructor`, which is the constructor to set the initial game board state to:
     ```
     1 2 3
     4 5 6
     7 8 9
     ```
   - `playerMove(int position, Player p)`, which is used to indicate that Player p wants to mark a position on the board. *Notes: This position may already be taken. If it was already taken, you need to not let this move happen. If this move is valid, then update the game board to indicate the player's added position.* For example, if Player X wants to move to position 1, the new board would look like:
     ```
     X 2 3
     4 5 6
     7 8 9
     ```

- `checkForWinner()`, which is used to check for a winner on the current game board. Winners can be Player X, Player O, C (for Cat), or I (for an unfinished, incomplete game).
- `toString()`, which shows the current board state. (You can also "showBoard" which basically calls the toString() method, but is a more meaningful method name.)
- Any other methods that you feel that you need to use.

3. Class `TicTacToeGameDriver`

This class has a main method, and static methods that control the game.

- Allow the user to play multiple games.
- For each game, start with an initial GameBoard. Player X makes the first move. Keep making alternating Player X/ Player O moves on the GameBoard until a winner (or cat game) is found. Output the winner.
- Keep track of the games that both Player X and Player O have won.
- Print out the ending totals.

Bonus Options:

1. Ask the user if they want 1 or 2 players. If they choose one player, then make Player O be an automated player and ask which level they want, beginner or advanced. If they chose the beginner level, the automated player randomly moves to an open position. If they chose the advanced level, the automated player plays to win. **10 points bonus possible.**

2. Instead of using the "System.out" console, use advance GUI features (see chapters 7 & 13, which we will cover later**.) Up to 10 points bonus possible.**

3. Add your own creative touches (or creative rules) that go above and beyond the basic assignment. **Various bonus points possible.**

Submission instructions:

You must submit your work through black board. Submit only the .java files. All source code files must be zipped into a single zip file with the file name in the following format (notice that it is all **LOWER CASE**):

assign_1_*xxxxxx*.zip where *xxxxxx* is your UNK username.

When this zip file in extracted, it should form a subfolder of the name: assign_1_*xxxxxx* with the source files under this subfolder. No other subfolder is allowed in the folder. Do not include the full development environment. If you do not remember how to do this, then ask.

Program Correctness.			
Player object class	**4** **4** **4**	constructors setters/getters toString and any other methods	
GameBoard class	**4** **10** **10** **4**	constructors playerMove checkForWinner toString and any other methods	
DriverClass	**7** **7** **6**	Allows user to play multiple games Single game functions correctly Track of number of wins per player	
	10	Display was neat and understandable.	
Questions 1-13	**30**		
Programming style			
	-5	Program was incorrectly named (wrong case). **TicTacToe**	
	-5	The opening message did not explains the purpose of the program and provides good instruction to the user.	
	-5	There are typos in user prompts or display.	
	-5	Message or code not neatly formatted or output was one long line of type across the screen.	
	-10	Program incorrectly zipped or named (see instructions) **assign_1_*xxxxxx*.zip**	
	-5	There is no message indicating that the program has terminated	
	-5	Inconsistent indentation	
	-5	Poor use of white space (too many or too few blank lines)	
	-5	Names not in source code	
	-5	Purpose of program not in source code	
	-5	Date file last modified not is source code	
	-5	Poor variable names	
	-20	Less than three helper methods used within code	
	-10	Failure to cite references/source code, if any was used	
Extra Credit	**10**	Allow for a variable number of players (20 points)	
	10	Allow for two jokers in the deck	
	5	Allow for customizing the names of the players (user would enter these)	
	10	Track high hands	
	10	Use of a three dimensional array to track all hands	