Lucky Vending Machine

Due Date : 12noon on Friday in Week 7 (8th Sept 2017)

Introduction

This assignment is worth 15% of the marks for your final assessment in this unit. **Heavy penalties will apply for late submissions.** This is an individual assignment and must be entirely your own work. You must attribute the source of any part of your code which you have not written yourself. Please note the section on plagiarism in this document.

The assignment must be done using the BlueJ environment.

The Java source code for this assignment must be implemented according to the ***Java Coding Standards*** for this unit.

Any points needing clarification must be discussed with your tutor in the tutorial class. Your tutor is your "client" - so you need to implement your program to suit his requirements.

Specification

For this assignment you will write a program that plays a rather simplistic ***Lucky Vending Machine***. This section specifies the required functionality of the program. ***Only a text interface is required for this program***; however, more marks will be gained for a game that is easy to follow with clear information/error messages to the player.

The aim of the ***Lucky Vending Machine*** is to simulate a physical vending machine commonly found in shopping malls, where a player inserts some coins, pulls a handle and wins a random prize (or no prize). You program will display a menu which allows the player to select various options to simulate the various vending machine operations. The "prizes" are decided by the computer (the vending machine) generating a random number between 1-5, with each number representing a particular prize (see prize table in the ***Program Logic*** section below). To "win" a prize, the player will need to guess correctly the number generated by the computer. The prizes are worth between $10-50, and each guess will cost the player between $1-5.

Each time the player chooses to play (ie. doing a "Guess A Prize"), he** is charged between $1-5, depending on what he wants to win (see **Prize Table**). The program will keep a record of all the prizes he has won, the total worth of the prizes, and how much money he has spent. At any point, the program can print out these statistics.

For this assignment, the program will only handle *ONE* player at any one time. However, the player can be changed while the program is executing.

** *for the rest of the document, "he" will be taken to mean "he/she".*

Program Logic

The *Lucky Vending Machine* begins with a welcome message followed by a menu with the following options :

```
Welcome to the Lucky Vending Machine
============================
(1) Set Up New Player
(2) Guess A Prize
(3) What Have I Won So Far?
(4) Display Game Help
(5) Exit Game
Choose an option :
```

Option (1) asks the user to enter a name for the "player". The player's name must not be blank. If this option is chosen again after a player has already been set up, a "new" player is created (ie. with a new name, no prizes, $0 spent). Note that the "new" player replaces the "old" player – ie. there is only ever one player at any one time.

Option (2) simulates a lucky guess operation. The computer generates a random number between 1-5. It then asks the player what prize he would like to win by trying to guess that number. The rules are :

Number Generated	Price Won	Price Worth	Cost to player
1	Pen	$10	$1
2	Book	$20	$2
3	DVD	$30	$3
4	Mouse	$40	$4
5	Keyboard	$50	$5

Prize Table

If the player guesses wrongly, he wins no prize and loses the money ($1-5).

For the purpose of this assignment, the "prizes" are simply implemented as **`Strings`**.

After each play, the computer records the player's prize/worth/spending.

If the user enters a number which is less than 1, or more than 5, it should be rejected, and he does not lose any money.

Option (3) displays some statistics about the current player's prizes (and their worths) and total amount spent.

Option (4) displays some brief instructions regarding game play. It must at least inform the player of the rules of play (see prize table under **Option (2)** above)

Option (5) exits the program. All player statistics should be cleared.

Additional Notes :

The menu must be displayed repeatedly, until the user chooses **Option (5)**. Inputs other than 1-5 should be rejected, and an error message printed.

If the user chooses **Options (2)** or **(3)**, before a player has been set up, an appropriate error message should be printed.

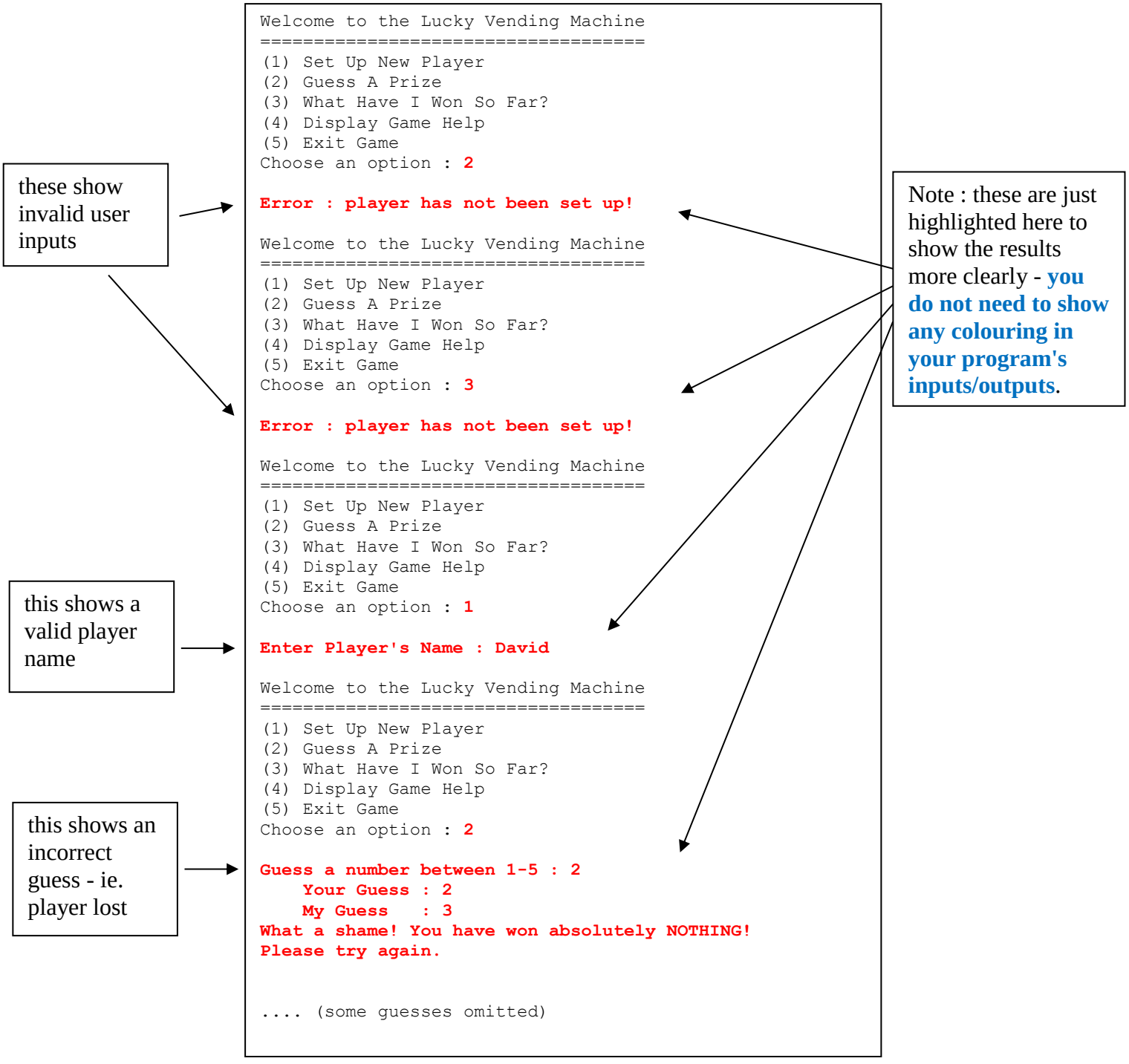
Your program must deal with invalid values entered by the user in a sensible manner.

For all the options, the inputs/outputs can be formatted in many different ways. Discuss with your tutor regarding how you should implement these in your program.

Assume that the user inputs are always of the correct data types (ie. when an integer is required, only an integer is entered, etc).

The sample screenshots above are meant to be examples. Your user interface need not be exactly as shown in those examples. However, you should discuss with your tutor about what to include in your interface.

Sample screenshots of typical inputs/outputs (including invalid inputs) for **Options (1), (2) & (3)** :



Sample screenshots continued…

this shows a correct guess - ie. player won

this shows an invalid menu choice

```
.... (some guesses omitted)


Welcome to the Lucky Vending Machine
====================================
(1) Set Up New Player
(2) Guess A Prize
(3) What Have I Won So Far?
(4) Display Game Help
(5) Exit Game
Choose an option : 2

Guess a number between 1-5 : 1
    Your Guess : 1
    My Guess   : 1
Congratulations! You have won a PEN, worth $10.

Welcome to the Lucky Vending Machine
====================================
(1) Set Up New Player
(2) Guess A Prize
(3) What Have I Won So Far?
(4) Display Game Help
(5) Exit Game
Choose an option : 6

Error : only choose 1-5


Welcome to the Lucky Vending Machine
====================================
(1) Set Up New Player
(2) Guess A Prize
(3) What Have I Won So Far?
(4) Display Game Help
(5) Exit Game
Choose an option : 2

Guess a number between 1-5 : 2
    Your Guess : 2
    My Guess   : 2
Congratulations! You have won a BOOK, worth $20.


.... (some guesses omitted)
```

Sample screenshots continued…

```
.... (some guesses omitted)


Welcome to the Lucky Vending Machine
====================================
(1) Set Up New Player
(2) Guess A Prize
(3) What Have I Won So Far?
(4) Display Game Help
(5) Exit Game
Choose an option : 2

Guess a number between 1-5 : 4
    Your Guess : 4
    My Guess   : 4
Congratulations! You have won a MOUSE, worth $40.

Welcome to the Lucky Vending Machine
====================================
(1) Set Up New Player
(2) Guess A Prize
(3) What Have I Won So Far?
(4) Display Game Help
(5) Exit Game
Choose an option : 3

Player David has won these prizes :
     PEN Book Mouse
worth a total $70
Total amount spent is $48

Welcome to the Lucky Vending Machine
====================================
(1) Set Up New Player
(2) Guess A Prize
(3) What Have I Won So Far?
(4) Display Game Help
(5) Exit Game
Choose an option : 5

Goodbye. Thank you for playing.
```

this shows some statistics for the player

NB : all the above screenshots are related (ie. should be read from top to bottom), with some guesses omitted for brevity.

Program Design

Your program must demonstrate your understanding of the object-oriented concepts and general programming constructs presented in the unit.

The data type of each field in the classes must be chosen carefully in accordance to the requirements of the program described in the preceding sections, and you must be able to justify the choice of the data type of the fields. You may want to include comments in the class to explain any assumption made.

Basic validation of values for fields and local variables should also be implemented. You should not allow an object of a class to be set to an invalid state. You should include appropriate constructor(s) for each class, and accessor/mutator methods for all the attributes.

You class design should include at least these classes (or some similar classes) :

`Game`

`Player`

`LuckyGuessGenerator`

Discuss with your tutor what attributes/behaviours are suitable for these classes, and how they interact with each other.

You may make the following assumptions :

- a `Game` object will be responsible for displaying the menus, accepting user responses, and performing the requested operations. It will make use of 1 `Player` object and 1 `LuckyGuessGenerator` object.

- a `Player` object will remember his own name, what prizes he has won (& their total worth), and how much he has spent. For the purpose of this assignment, you can just store all the prizes as a string containing all their names, separated by spaces,

  eg. "`Book DVD Mouse Book DVD Book`" - these mean he has won 6 prizes so far

  If you wish to implement the prizes in a different way, please seek advice from your tutor first.

- a `LuckyGuessGenerator` object will be used to generate a number between 1-5. This is used by the program to generate the random number for the player to guess.

Assessment

Assessment for this assignment will be done via an ***interview*** with your tutor. The marks will be allocated as follows:

- Object-oriented ***design*** quality. This will be assessed on appropriate implementation of classes, fields, constructors, methods and validation of object's state : **30%**
- Adherence to FIT9131 Java coding standards : **10%**
- Program functionality in accordance the requirements : **60%**

You must submit your work by the submission deadline on the due date (a late penalty of 20% per day, inclusive of weekends, of the possible marks will apply - up to a maximum of 100%). ***There will be no extensions*** - so start working on it early.

Marks will be deducted for untidy/incomplete submissions, and non-conformances to the *FIT9131 Java Coding Standards*.

Interview

This is where most of your marks will come from.

You will be asked to demonstrate your program at an "***interview***" following the submission date. At the interview, you will be asked to explain your code, your design, discuss design decisions and alternatives, and modify your code as required. ***Marks will not be awarded for any section of code or functionality that you cannot explain satisfactorily*** (your tutor may also delete excessive in-code comments before you are asked to explain that code).

In other words, ***you will be assessed on your understanding of the code***, and not solely on the actual code itself.

Interview times will be arranged in the tutorial labs in *Week 7*. It is your responsibility to attend the lab and arrange an interview time with your tutor. ***Any student who does not attend an interview will receive a mark of 0 for the assignment.***

For the on-campus students, the actual interviews will take place during the normal lab times in *Week 8*. Your own tutor will perform the interviews.

Submission Requirements

The assignment must be uploaded to ***Moodle*** on or before the due date (as listed at the start of this document). The link to upload the assignment will be made available, along with a check-list of things to submit, in the ***Assignments*** section of the unit's Moodle site before the submission deadline. The submission requirements are as follows:

A .zip file uploaded to Moodle containing the following components:

- the **BlueJ** project you created to implement your assignment. The .zip file should be named with your Student ID Number. For example, if your id is **12345678**, then the file should be named **12345678_A1.zip**. Do not name your zip file with any other name.
    - it is your responsibility to check that your ZIP file contains all the correct files, and is not corrupted, before you submit it. If your tutor cannot open your zip file, or if it does not contain the correct files, you will not be assessed.
- a completed **Assignment Cover Sheet**. This will be available for download from the unit's Moodle site before the submission deadline. You simply complete the editable sections of the document, save it, and include it in your .zip file for submission.

Marks will be deducted for failure to comply with any of these requirements.

Warning : **there will be no extensions to the due date**. Any late submission will incur the 20% per day penalty. It is strongly suggested that you submit the assignment well before the deadline, in case there are some unexpected complications on the day (eg. interruptions to your home internet connection).

Plagiarism

Cheating and plagiarism are viewed as serious offences. In cases where cheating has been confirmed, students have been severely penalised, from losing all marks for an assignment, to facing disciplinary action at the Faculty level. Monash has several policies in relation to these offences and it is your responsibility to acquaint yourself with these.

Plagiarism (http://www.policy.monash.edu/policybank/academic/education/conduct/plagiarism-policy.html)

Discipline: Student Policy (http://www.policy.monash.edu/policy-bank/academic/education/conduct/student-discipline-policy.html