Midterm Exam, Faculty of Engineering, Chulalongkorn University
Course ID: 2110215, Course Name: Programming Methodology
First Semester, Date: September 29, 2015Time: 8:30-11:30

Name ..... Student ID. ..... No. in CR .....

Repoints

1. The exam duration is 180 minutes.
2. Write your Student ID, full name, and your number in CR58 in the space provided on the top of this page.
3. Your answer must only be written on this exam paper.
4. Documents, calculators, or computers are allowed inside the exam room; however, internet is prohibited.
5. Borrowing is not allowed unless it is supervised by the proctor.
6. ***** You must not bring any part of this exam paper outside. The exam paper is a government's property. Violators will be prosecuted under a criminal court. *****
7. Students, who wish to leave the exam room before the end of the exam period, must raise their hands and ask for permission before leaving the room. Students must leave the room in the orderly manner.
8. Once the time is expired, student must stop writing and must remain seated quietly until the proctors collect all the exam papers or given exam booklets. Only then, the students will be allowed to leave the room in the orderly manner.
9. Any student who does not obey the regulations listed above will receive punishment under the Faculty of Engineering Official Announcement on January 6, 2003 regarding the exam regulations.
    a. With implicit evidence or showing intention for cheating, student will receive an F in that subject and will receive an academic suspension for 1 semester.
    b. With explicit evidence for cheating, student will receive an F in that subject and will receive an academic suspension for 1 year.

Please sign and submit

Signature (……)

1 Objective

- Design the basic principles of program design with objects and classes (OOP) including setters & getters, constructor, access modifiers, etc.
- Be able to apply the inheritance concept (*extends*) and interface (*implements*)
- Be able to apply the polymorphism concept: generic methods and arrays
- Understand the class UML diagram and be able to implement java project in Eclipse
- Be able to use JUnit Test Framework in order to make a program without errors

2 Problem Statement: Chocobo Racing

In this game, a player tries to simulate the racing game that runners come from different games. There are three runners:

- RunnerChocobo (from Final Fantasy Game)
- JumperChocobo (from Final Fantasy Game)
- Pikachu (from Pokemon Game)

  The followings are their statuses:
  - Runner Chocobo
    - Speed : 3m/turn
  - Jumper Chocobo
    - Speed : 7.5m/2 turns
  - Pikachu
    - Speed : 4m/turn
- At start of this racing simulation game, all characters start at the distance 0 and then run with their speed.
- At start of each turn, the game manager will arrange the order for each character to run according to their distance. The character with the lowest distance gets to run first. (In our simulation, characters do not move at the same time.) If two characters have the same distance, the character with highest priority gets to run first. The priority are Runner Chocobo > Jumper Chocobo > Pikachu.
- Unfortunately, while the racing simulation game is running, our runners may face some obstruction, causing slow-down/or temporary stop. Pikachu and Jumper Chocobo can be affected by such obstruction, but Runner Chocobo can dodge the obstruction.
- The game ends when a character reaches the goal at distance 100 m.

3 Instruction

- **At the beginning**
- Set your workspace to “C:\temp\{StudentID}”
- Create project “2110215_Midterm2015_{StudentID}_{FirstName}”
- **At the end (how to submit your work)**
- Create runnable jar file and place it at the root directory of your project
- Close Eclipse
- DO NOT turn off your computer and DO NOT take the exam or any files out of the lab.

4 Implementation Details

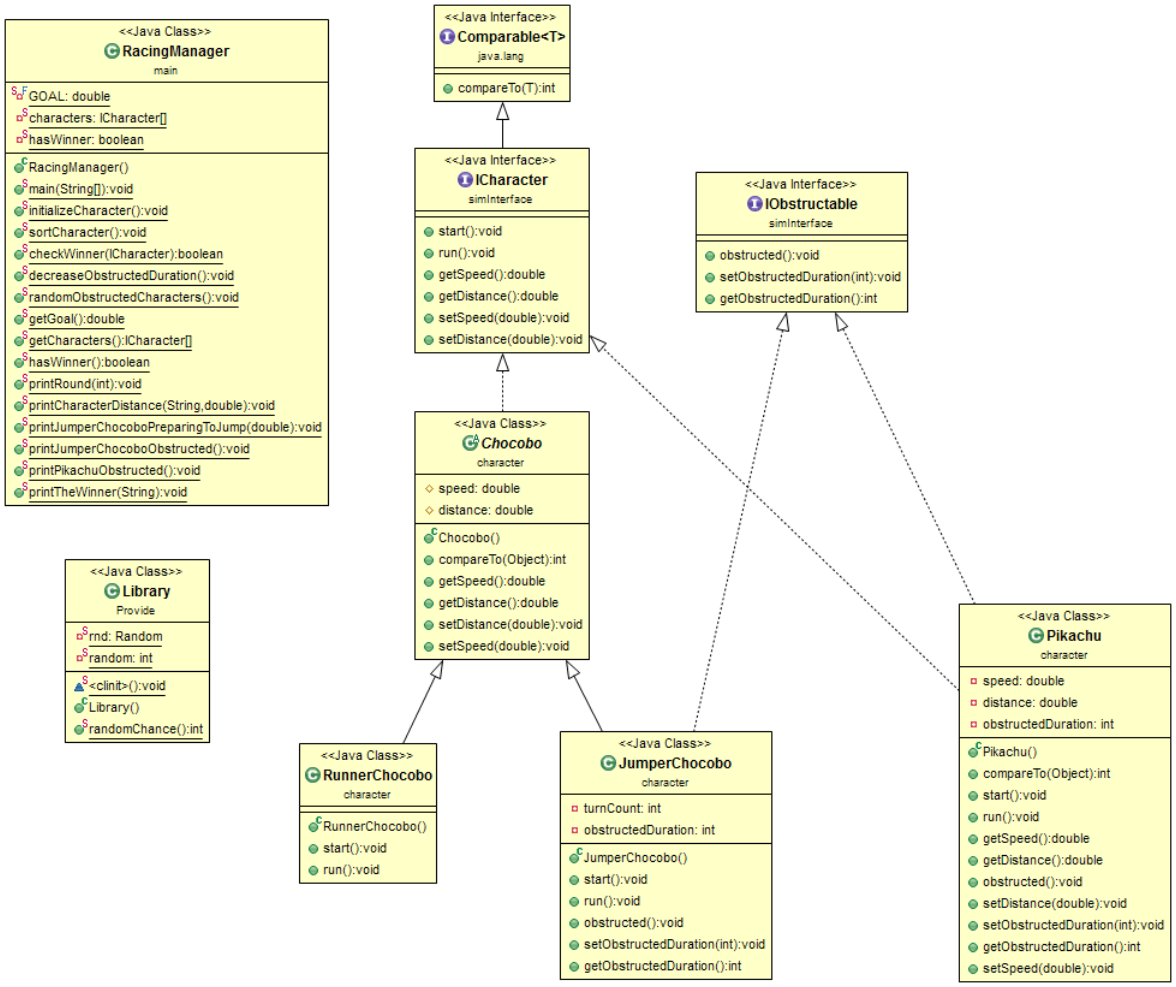

Figure illustrates the UML class diagram of the program. The class-package structure is summarized below:

- Package “main”: RacingManager
- Package “character”: Chocobo, JumperChocobo, RunnerChocobo, Pikachu
- Package “simInterface”: ICharacter, IObstructable
- Package “test”: Junit Test Case
- Package “Provide”: Library

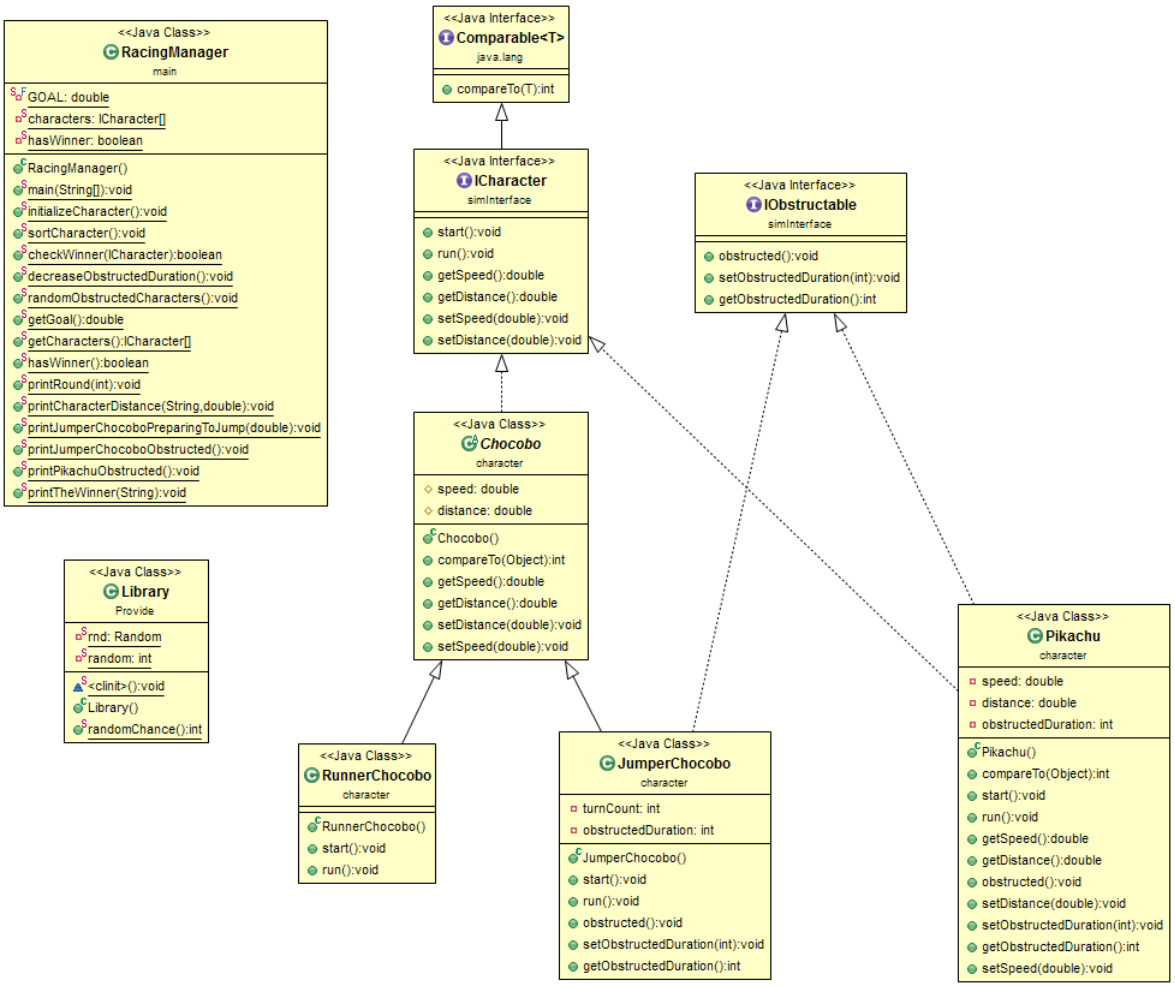

Figure 1. The class diagram of the program.

*** Access modifiers and relationships must follow the UML diagram ***

4.1 Interface "IObstructable"

The classes that can be obstructed must implement this interface.

4.1.1 Method

- + void obstructed(); It is an action that is called by *RacingManager* in order to trigger an obstruction to a runner that implements this interface.
- + void setObstructedDuration(int); assign obstructed duration to this character
- + int getObstructedDuration(); return a value representing remaining turns that the character is obstructed

4.2 Interface "ICharacter"

Runner classes must implement this interface.

4.2.1 Method

- + void start(); It is an action to prepare this character for the run.
- + void run(); It is an action performed when this character is chosen to run by *RacingManager*.
- + double getSpeed(); return a value representing the current speed of this character.
- + double getDistance(); return a value representing the current distance of this character.
- + void setSpeed(double speed); assign speed to this character.
- + void setDistance(double distance); assign distance to this character.

4.3 Abstract Class "Chocobo"

It is a **template** class representing each Chocobo.

4.3.1 Field

- # double speed; current speed of this Chocobo
- # double distance; current distance of this Chocobo

4.3.2 Method

- + double getSpeed(); return a value representing the current speed of this character.
- + double getDistance(); return current distance of this Chocobo.
- + void setDistance(double distance); This method required for JUnit test case to set distance to the character.
- + void setSpeed(double speed); assign speed to this character.
- + int compareTo(Object o); this method compares 2 characters and decide which one gets to run first. The return value is as the following:
    - 1 :if this character gets to run after the object o.
    - -1 :if this character gets to run before the object o.
    - Running order
        - A character with higher distance gets to run after a character with lower distance.
        - if both characters have same distance then use priority instead.
            - Priority : RunnerChocobo >JumperChocobo > Pikachu

- **Note1** : **DO NOT** compare characters using their speed. You MUST compare them by using **instanceof**.
- **Note2** : Operator "==" cannot be used to compare **double** values, use **Double.compare(Double d1, Double d2)** instead.

4.4 Class "RunnerChocobo"

This class is inherited from Chocobo.

4.4.1 Constructor

- + RunnerChocobo(); this Chocobo runs at the speed of 3 m/turn.

4.4.2 Method

- + void start(); It set the distance of this character to be 0.
- + void run(); It is a method to run this character by adding the current distance by its current speed.

4.5 Class "JumperChocobo"

This class extends from Chocobo and can be obstructed (Obstructable). JumperChocobo does not run every turn; it just runs every 2 turns. Its first turn is used to prepare to jump and its second turn is used to jump.

4.5.1 Field

- - int turnCount; turn counter used to determine if this character should jump or not.
- - int obstructedDuration; the number of remaining turns that the character is obstructed.
    1. If it is 0, the character is not obstructed.

4.5.2 Constructor

- + JumperChocobo(); This Chocobo runs at the speed of 7.5 m on every other turn (7.5 m/2 turns). Also, it is not obstructed when it is created.

4.5.3 Method

- + int getObstructedDuration(); return current *obstructedDuration*.
- + void setObstructedDuration(); assign *obstructedDuration* to this character.
- + void start(); It set the distance of this character to be 0.
- + void run(); add the current distance by its current speed every 2 turns.
    - **Note**: this method is called every turn by *RacingManager*. However, this character runs every other turn (2 turns).You can achieve this by updating and checking *turnCount* variable.
- + void obstructed(); decrease the speed of this character by half. The obstructed effect lasts for 2 turns.
    - If the character is already obstructed when this method is called, this method does nothing.

4.6 Class "Pikachu"

This class is a character and can be obstructed (IObstructable). This character has the highest speed.

4.6.1 Field

- - double speed; current speed of this character.
- - double distance; current distance of this character.
- - int obstructedDuration; the number of remaining turns that the character is obstructed.
    1. If it is 0, the character is not obstructed.

4.6.2 Constructor

- + Pikachu(); initialize its speed to 4 m/turn. Also, it is not obstructed when it is created.

4.6.3 Method

- + double getSpeed(); return current speed of Pikachu.
- + double getDistance(); return current distance of Pikachu.
- + void setSpeed(double speed); assign speed to this character.
- + void setDistance(double distance); This method required for JUnit test case to set distance to the character.
- + int getObstructedDuration(); return current *obstructedDuration*.
- + void setObstructedDuration(int obstructedDuration); required for JUnit test case to set *obstructedDuration* value.
- + int compareTo(); exactly the same as in section **4.3.2**.
- + void start(); It set the distance of this character to be 0.
- + void run(); add the current distance by its current speed.
- + void obstructed(); decrease the speed of this character to 0. The obstruction effect lasts for 1 turn.
    - If the character is already obstructed when this method is called, this method does nothing.

4.7 Class "RacingManager"

4.7.1 Field

- + double GOAL; how far the goal is (in meter), the default value is 100.00.
- + ICharacter characters[]; this variables used to store all runners.
- + boolean hasWinner; a flag that is true when there is a winner. The default value is false.

4.7.2 Method

- + double getGoal(); return the GOAL distance.
- + ICharacter[] getCharacters(); return all characters in *characters[].*
- + Boolean hasWinner(); return a *winner* flag.
- + void initializeCharacter(); initialize *characters[]* with 3 runners in this simulation(RunnerChocobo, JumperChocobo and Pikachu and move all of them to the starting point (distance = 0).
- + void sortCharacter();sort all characters in *characters[]* using Comparable to arrange running order.

    - The first character (index 0) in the sorted array is the first character to run in the current turn.
- + Boolean checkWinner(ICharactercrt); this method is used after each character runs to check if the character reaches the goal.
    - If it is, then sets the winner flag and returns true. Otherwise, returns false.
- + void randomObstructedCharacters();random chance to obstruct JumperChocobo or Pikachu or both before running each turn. You **MUST** use the provided library as described in section 3.8 (Provide.*Library.randomChance()*) to get a random number from 1 to 100. It **MUST** be called **only once** in this method.
    - 20% chance to obstruct JumperChocobo
    - 20% chance to obstruct Pikachu
    - 20% chance to obstruct both JumperChocobo and Pikachu
    - 40% chance of no obstruction.
- + void decreaseObstructedDuration(); this method decreases each obstructable character's obstructedDuration by one every turn if the character is obstructed. If obstructedDuration is 0 then set characters' speed to their default speed.
- + void main(String[]); show welcome quote and then receive [ENTER] from user to start simulation. When the simulation starts, all characters are initialized. Then, each turn is simulated as the following:
    1) Sort characters' running order for the current turn.
    2) Decrease *obstructedDuration* for obstructed character(s).
    3) Randomly obstruct character(s).
    4) Assign **each character** to run and check whether it is the winner
    5) If there is a winner, end the simulation.
- You should use following methods to print out console to get the same result as **ChocoboRacing.exe**
    1) + void printRound(int roundCount); use this method to print console for each round. The parameter *roundCount* specifies current round number.
    2) + void printCharacterDIstance(String characterName, double characterDistance); use this method to print characters' distance for each turn.
    3) + void printJumperChocoboPreparingToJump(double jumperChocoboDistance); use this method to print JumperChocobo distance when it is preparing to jump.
    4) + void printJumperChocoboObstructed(); use this method to print event when JumperChocobo is obstructed.
    5) + void printPikachuObstructed(); use this method to print the event when Pikachu is obstructed.
    6) + void printTheWinner(String characterName); use this method to print the event when there is a winner.

4.8 Package "Provide" Class "Library"

4.8.1 Method

- + int randomChance(); return random chance between 1 to 100 with fixed seed.
    1. In **ChocoboRacing.exe**, we used

- 1 – 20 to obstruct JumperChocobo,
- 21 – 40 to obstruct Pikachu,
- 41 – 60 to obstruct both JumperChocobo and Pikachu, and
- 61 – 100 for otherwise.

5 JUnit Test Case

- There are four provided test cases including:
    1. TestRunnerChocobo
    2. TestJumperChocobo
    3. TestPikachu (incomplete)
    4. TestRacingManager
- You **MUST** finish JUnit Test Case for TestPikachu.

6 ObjectAid UML Class Diagram

- Generate UML Class diagram using ObjecAid in Eclipse.
    1. Generate all implemented classes same as Figure 1, including Java interface and external library.
    2. Save both ObjectAid ucls "**ChocoboRacing.ucls**" file and image as "**ChocoboRacing.png**" in project root directory.

Example output log fromChocobo Racing Game

(Your result should be THE SAME AS the output below)

```
Welcome to CHOCOBO RACING.
The goal at distance 100 m.

Press 'any key' to START.

Initialized Characters
- RunnerChocobodistance : 0.0
- JumperChocobodistance : 0.0
- Pikachu distance : 0.0

Round 1
+ JumperChocobo has obstructed > speed down
for 2 seconds
+ Pikachu has obstructed > stop running for
1 second
- RunnerChocobodistance : 3.0
- JumperChocobodistance : 0.0 and preparing
to Jump
- Pikachu distance : 0.0

Round 2
JumperChocobodistance : 3.75
Pikachu distance : 4.0
RunnerChocobodistance : 6.0

Round 3
+ JumperChocobo has obstructed > speed down
for 2 seconds
+ Pikachu has obstructed > stop running for
1 second
- JumperChocobodistance : 3.75 and
preparing to Jump
- Pikachu distance : 4.0
- RunnerChocobodistance : 9.0

Round 4
+ Pikachu has obstructed > stop running for
1 second
- JumperChocobodistance : 7.5
- Pikachu distance : 4.0
- RunnerChocobodistance : 12.0

Round 5
+ Pikachu has obstructed > stop running for
1 second
+ JumperChocobo has obstructed > speed down
for 2 seconds
- Pikachu distance : 4.0
- JumperChocobodistance : 7.5 and preparing
to Jump
- RunnerChocobodistance : 15.0

Round 6
- Pikachu distance : 8.0
- JumperChocobodistance : 11.25
- RunnerChocobodistance : 18.0

                • • •

Round 27
+ Pikachu has obstructed > stop running for
1 second
- JumperChocobodistance : 63.75 and
preparing to Jump
- Pikachu distance : 72.0
- RunnerChocobodistance : 81.0

Round 28
+ JumperChocobo has obstructed > speed down
for 2 seconds
+ Pikachu has obstructed > stop running for
1 second
- JumperChocobodistance : 67.5
- Pikachu distance : 72.0
- RunnerChocobodistance : 84.0

Round 29
- JumperChocobodistance : 67.5 and
preparing to Jump
- Pikachu distance : 76.0
- RunnerChocobodistance : 87.0

Round 30
+ Pikachu has obstructed > stop running for
1 second
- JumperChocobodistance : 75.0
- Pikachu distance : 76.0
- RunnerChocobodistance : 90.0

Round 31
+ Pikachu has obstructed > stop running for
1 second
- JumperChocobodistance : 75.0 and
preparing to Jump
- Pikachu distance : 76.0
- RunnerChocobodistance : 93.0

Round 32
- JumperChocobodistance : 82.5
- Pikachu distance : 80.0
- RunnerChocobodistance : 96.0

Round 33
+ JumperChocobo has obstructed > speed down
for 2 seconds
- Pikachu distance : 84.0
- JumperChocobodistance : 82.5 and
preparing to Jump
- RunnerChocobodistance : 99.0

Round 34
+ Pikachu has obstructed > stop running for
1 second
- JumperChocobodistance : 86.25
- Pikachu distance : 84.0
- RunnerChocobodistance : 102.0

RunnerChocobo is the WINNER !!!
```