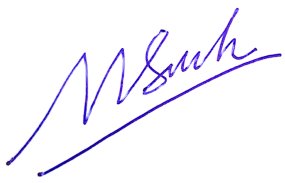
International University - VNUHCM
School of Computer Science and Engineering

Final Examination

Date: 17/01/2022; Duration: 120 minutes

Online; open book exam

SUBJECT: Object-Oriented Programming (IT069IU)	
Approval Signature  Full name: Nguyễn Văn Sinh	Lecturer: Signature  Full name: Trần Thanh Tùng
Proctor 1 Signature Full name:	Proctor 2 Signature Full name:
STUDENT INFO	
Student name: **Student ID:**	

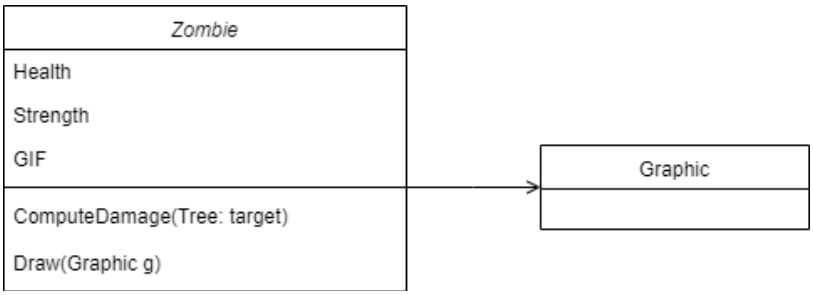
INSTRUCTIONS: the total of points is 100 (equivalent to 40% of the course)

1. *Purpose:*
   - Test your knowledge on object-oriented programming in the following topics: Classes, Objects, Encapsulation, Abstraction, Inheritance, Polymorphism, Generic, SOLID principles, and design patterns (CLO1, CLO3)
   - Examine your skill in analysis and design classes and algorithms (CLO2)
2. *Requirement:*
   - Read carefully each question and answer it following the requirements
   - Write the answers and draw models CLEAN and TIDY directly in the exam paper

   - **SUBMIT YOUR EXAM TO THE BLACKBOARD**

1. **(30 marks)** In Object-Oriented Programming,
   a. Describe the keyword "Abstract class" (15 marks).
   b. Give an example of a concrete class that extends from an abstract class (15 marks)

2. **(25 marks)** Given a design below



   a. Explain why the design is violating the Single Responsibility Principle (**15 marks**)?
   b. Redesign (by drawing a new diagram) to make it conform with the Single Responsibility Principle (**10 marks**)

3. **(30 marks)** Dependency Inversion Principle: high-level modules should not depend upon low level modules. Both should depend upon abstraction.

   The following code is an implementation of a login function for `MySocialNetwork`, but it violates the Dependency Inversion Principle.

   a. Explain why the code violates the Dependency Inversion Principle **(15 marks)**
   b. Rewrite the code to make it conform with the principle **(15 marks)**

```
public class My_HCMIU_Email {
  public void loginWithStudentID(String username, String password)  {
    /* ……..*/
  }
}

public class MySocialNetwork {
  private My_HCMIU_Email loginService;
  public void SetLoginService(My_HCMIU_Email value) {
    loginService = value;
  }
  public bool login(String username, String password) {
    return loginService.loginWithStudentID(username, password);
  }
}
```

4. **(15 marks)** Assuming you are developing a Plants vs Zombies game. In the game, there are many plants and many zombies.

```
public class Zombie {

  int nDestroyedPlants = 0;

  /*…*/

  // Is automatically called
  public void hasDestroyedAPlant() {

    nDestroyedPlants ++;

    /*…*/

  }

  public void freeze_2s() { /*…*/}

}
```

```
public class Plant {

  int nKilledZombies = 0;

  /*…*/

  // Is automatically called
  public void hasKilledAZombie() {

    nKilledZombies ++;

    /*…*/

  }

  public void freeze_1s() { /*…*/}

}
```

We want to implement a new feature named *Reward*:

- Whenever a plant kills its 50th zombie, all zombies currently on the board are frozen for 2 seconds.
- Whenever a zombie destroys its 20th plant, all plants are frozen for 1 second.

Use the Observer pattern to implement the Reward feature

a. Identify and implement/modify classes corresponding to the Subject and the Observer classes in the pattern (**10 marks**)
b. Implement a test function to create 2 plants, 2 zombies, any other objects if needed, and then setup the connections among all components following the Observer pattern (**5 marks**)

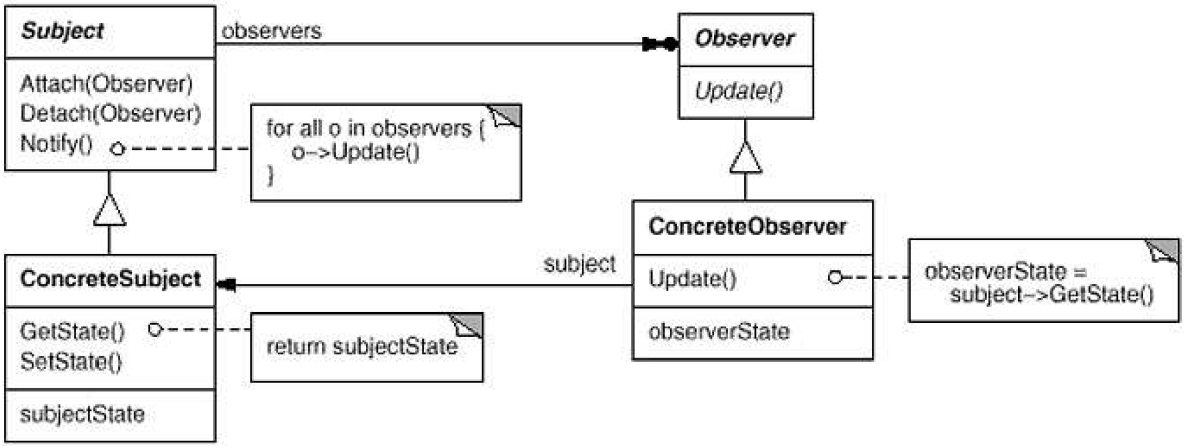


Figure 1 - Structure of the Observer pattern

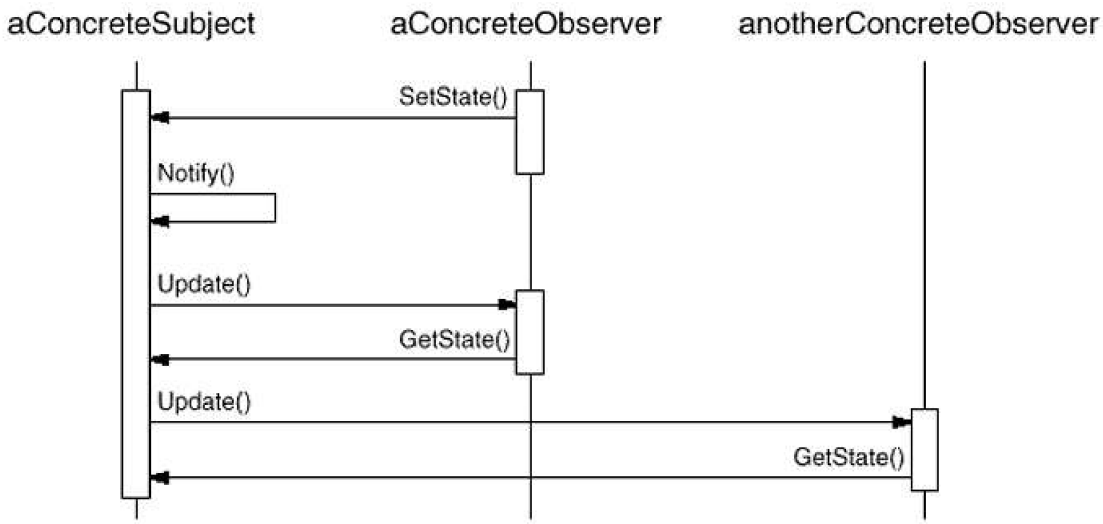


Figure 2 - Sequence diagram of an implementation of the Observer pattern

-- The end --