# EXAM 10 August 2015, questions

Software Design (City University of Hong Kong)

# CITY UNIVERSITY OF HONG KONG

Course code & title : CS3342 Software Design

Session : Semester B 2014/15

Time allowed : Two hours



This paper has **10** pages (including this cover page).

1. This paper consists of **6** questions.
2. Answer ALL questions within the examination booklet.

*This is a **closed-book** examination.*

*No materials or aids are allowed during the whole examination. If any unauthorized materials or aids are found on a candidate during the examination, the candidate will be subject to disciplinary action.*

**Student Number:** _____

**Programme:** _____

**Seat Number:** _____

| | *Answer all questions* | | | | | | |
|---|---|---|---|---|---|---|---|
| | *CILO 1* | *CILO 2* | *CILO 4* | *CILO 3* | *CILO 4* | *CILO 5* | |
| **Question** | **1** | **2** | **3** | **4** | **5** | **6** | **Total** |
| **Max** | **10** | **20** | **5** | **35** | **25** | **5** | **100** |
| | | | | | | | |

## Question 1 – Software Engineering Development Processes – CILO 1 (10 Marks):

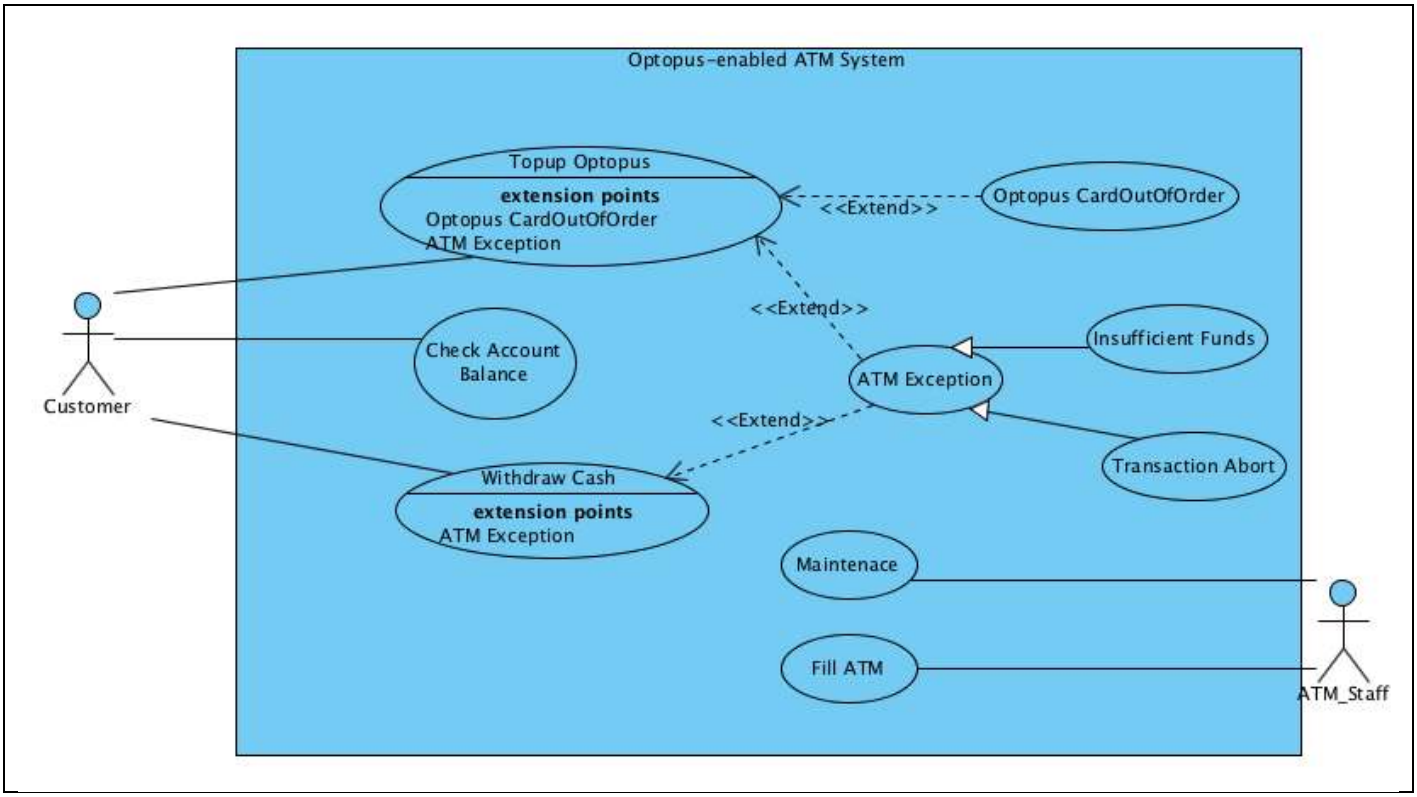
**Fill in the Terminology ID according to the list above (1 Mark Each, 10 Marks total):**

| Terminology ID | Description/Illustration |
|---|---|
| **7** | Reuse and integration is its core objective where systems are built from integrating existing commercial off the shelf components. |
| **9** | The study and an application of engineering to design, development, and maintenance of software |
| **5** | The _____ is easy to follow, it is structured which provide a template into which methods for analysis, design, code, and testing and maintenance can be utilized. Due to its sequential nature, it does not reflect reality, which relies heavily on completed requirements at start, and problems in the specification may be discovered very late during coding or integration stages. |
| **8** | _____ provides a service without regard to where the component is executing or its programming language. Their interface is normally published and all interactions are through the published interfaces to allow reuse and integrate with other systems. |
| **2** | The _____ provides the core functionalities first, and iteratively adds a new functionality in each release cycle until the product is completed. |
| **4** | The _____ builds on incremental model with emphases on short development cycle. It involves multiple teams, and assumes that the system can be modularized. It will fail we do not have strong and skillful teams. |
| **3** | Core requirements are somewhat understood but additional requirements are changing fast. Specification, development and validation activities are concurrently carried out with rapid feedback to customer across these activities, it is suitable for rapidly changing requirements, but time estimation and project completion date may be difficult to forecast. |
| **6** | It explicitly use the divide and conquer principle in the process, where each team works on its own allocated component in parallel with the other teams, and integrates these components periodically. |
| **1** | This provides a high level conceptual understanding of the system design, and useful platform to allow better communication with customer to identifying requirements expected. However it may be leading customer with incorrect perception about the final product. |
| **10** | All the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems following requirements specification and before programming. |

## 2. User Requirements Specification – (20 Marks) CILO 2

## Octopus-enabled ATM system

a) **(10 Marks)**




Correctly showing Octopus-enabled ATM system label : 1 Mark
Correctly showing Actors : 1 Mark
Correctly showing Topup/CheckBal/Withdraw use cases : 2 Marks
Correctly showing Maintenance and Fill ATM use cases : 2 Marks
Correctly showing all ATM exceptions : 4 Marks
(10 Marks Total)

b) **(10 Marks)**

| **Use Case Name:** | **Withdraw Cash** | |
|---|---|---|
| **Actor(s):** | Customer | |
| **Description:** | This use case describes the process of a customer completing Check Account Balance Process. | |
| **Reference ID:** | ATM-1.0 | |
| **Typical course of events:** | **Actor Action** | **System Response** |
| | **Step 1:** The Customer inserts his bankcard into the ATM.<br><br>**Step 3**: The Customer enters his correct PIN and confirms it.<br><br>**Step 5**: The Customer selects the "Withdraw Cash" function by pressing the appropriate button.<br><br>**Step 7**: The Customer enters the desirable amount to withdraw.<br><br>**Step 9**: The Customer collects Cash and removes the card. | **Step 2:** The ATM system shows the enter PIN screen to the Customer.<br><br>**Step 4**: The ATM welcomes the Customer and opens the main menu.<br><br>**Step 6:** The ATM system asks the Customer to enter an amount for cash withdraw.<br><br>**Step 8:** The ATM system processes the transaction request, and dispenses the cash. |
| **Alternative course of events:** | **Step 8a**: If the requested amount is < balance on the account, an Insufficient Funds use case will be used<br>**Step 8b**: If the user has pressed cancel button, a Transaction Abort use case procedure will be taken place. | |
| **Precondition:** | The ATM is working and customer is in front of the terminal. | |
| **Postcondition:** | The Customer can view/see his account balance. | |

Correctly showing Normal Events: : 5 Marks
Correctly showing Alternative course of events: : 3 Marks
Correctly showing Precondition/postcondition: : 2 Marks
(10 Marks Total)

## 3. Roles of Variables (5 Marks) – CILO 4

| Variable | Role of Variable (1 Mark Each) |
|---|---|
| i | Stepper |
| sum | Gatherer |
| squared | Transformation |
| input | Most Recent Holder |
| max | Constant |

## 4. Object Oriented Analysis (35 Marks) – CILO 3

4(a)

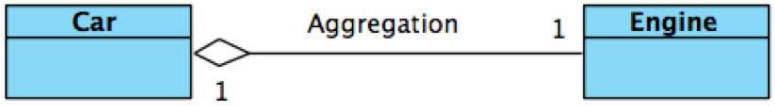
```
class Engine {
    public Engine() { };
    public void work() {
        System.out.println ("Engine Works...");
    }
}
```

**Complete the following diagram to use the correct UML notation and indicate their correct linkage relationship (Associations / Aggregation / Composition).**

```
class Car {
    private final Engine engine = new Engine();

    public void move() {
        engine.work();
    }
}
```

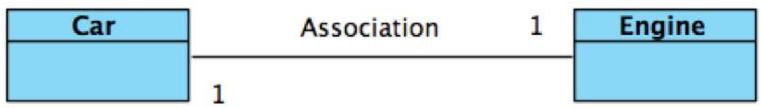

**This is _____ relationship.**

```
class Car {
    public void move(Engine engine){
       engine.work();
    }
}
```




**This is _____ relationship.**

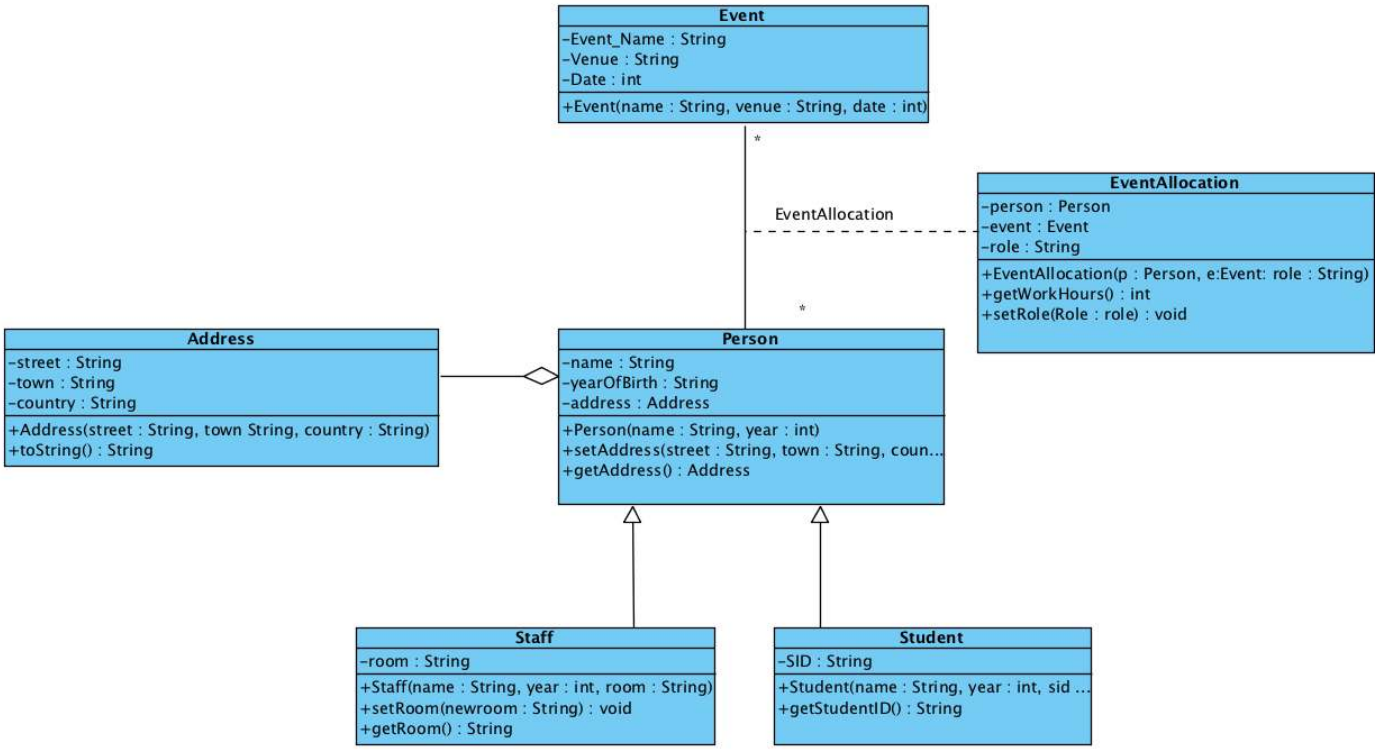
```
class Car {
    private Engine engine;

    public void installEngine (Engine engine) {
        this.engine = engine;
    }
    public void move() {
            engine.work();
    }
}
```




**This is _____ relationship.**

**[ Marking: 2 mark each correct linkage, up to 5 marks]**

4(b) Class Diagram:



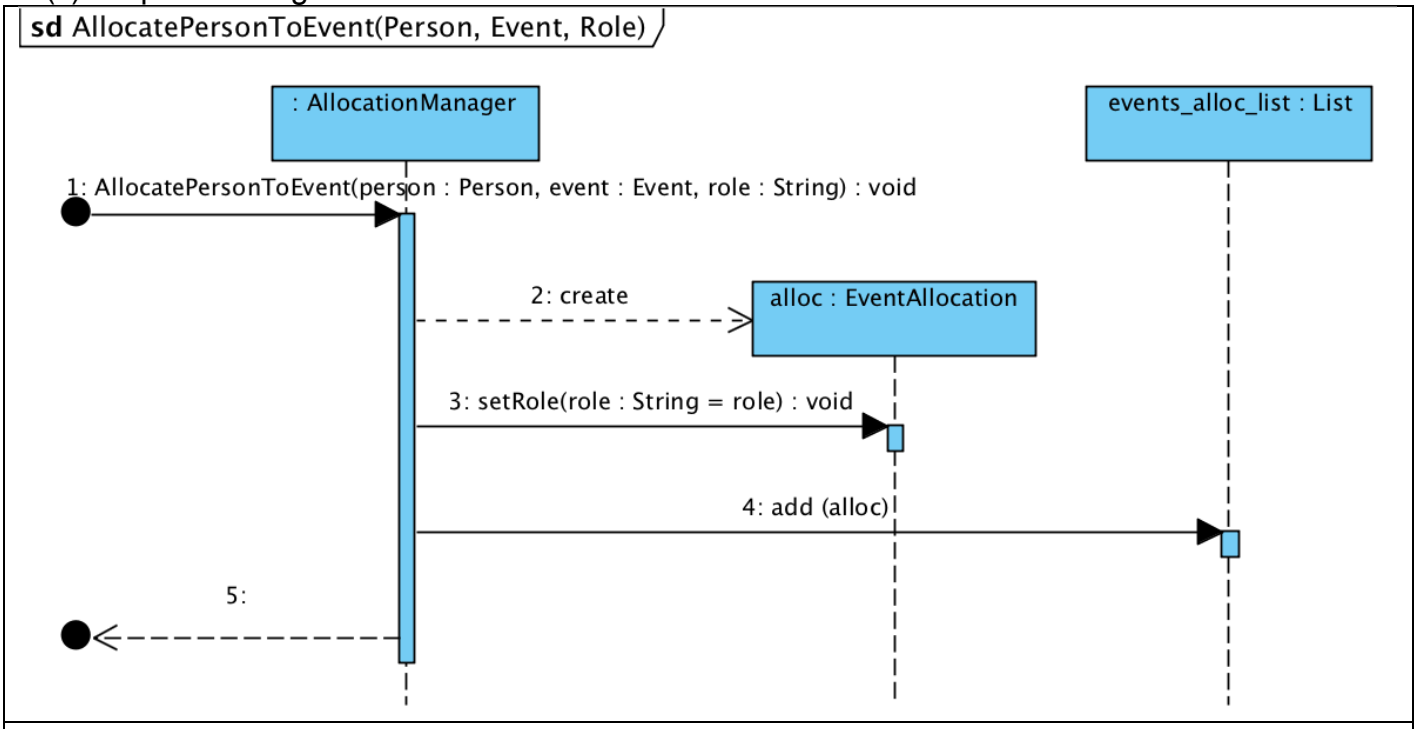

Part I (10 Marks)

Correctly showing classes Person, Staff, Student and Address :3 Marks
Correctly showing inheritance notations (Staff,Student) to Person :2 Marks
Correctly showing attribute/operation details within classes :3 Marks
Correctly showing aggregation between Person—Address :2 Marks

Part II (10 Marks)

Correctly showing Event class and attributes :3 Marks
Correctly showing Event Allocation attributes and operations :3 Marks
Correctly showing Associative Entity Relationship :3 Marks
Correctly showing Linkage between Event and Person :1 Mark

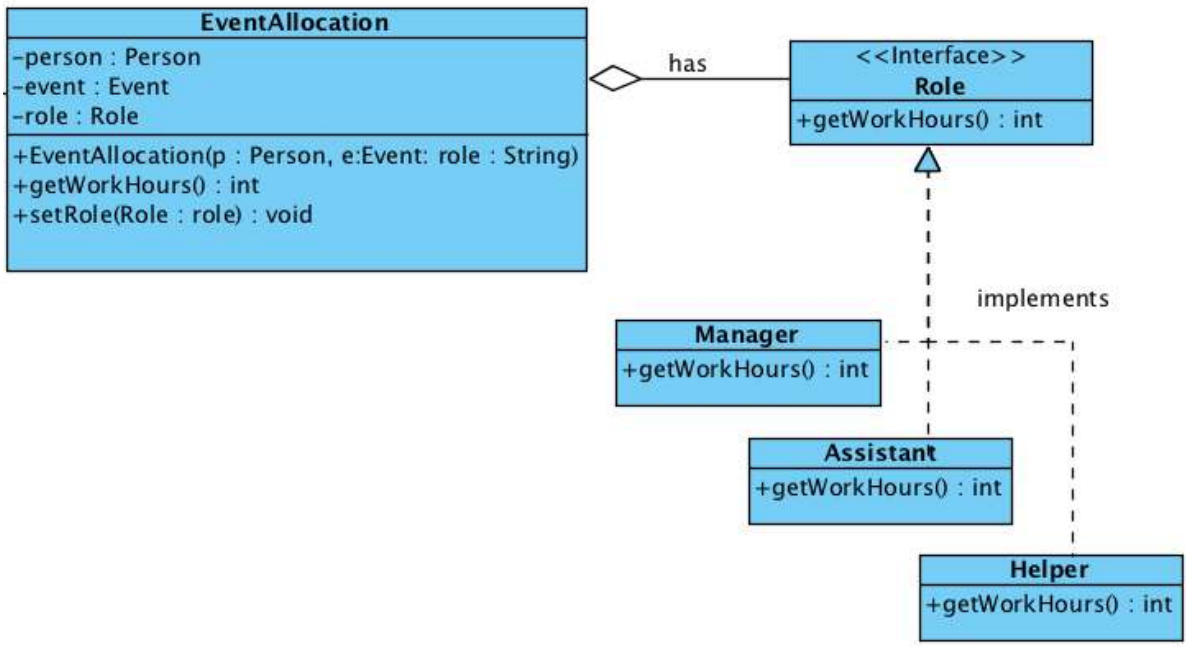
4(c) Sequence Diagram:




Correctly showing create alloc:EventAllocation : 3 Marks
Correctly showing setRole () under alloc:EventAllocation : 2 Marks
Correctly showing add (alloc) sequence : 2 Marks
Correctly showing events_alloc_list:List : 3 Marks

**5. Software Design Principles and Patterns (25 Marks) - CILO 4**

5(a)



EventAllocation:
- role : Role : 1 Mark
Role (State Pattern)
- correctly showing Role <Interface> and Manager/Assistant/Helper subclasses : 3 Marks
-
Correctly showing Aggregation between EventAllocation and Role : 2 Marks
Correctly showing dotted line / implements Role interface : 1 Mark
Correctly showing getWorkHours() in multiple parts : 3 Marks

5(b)

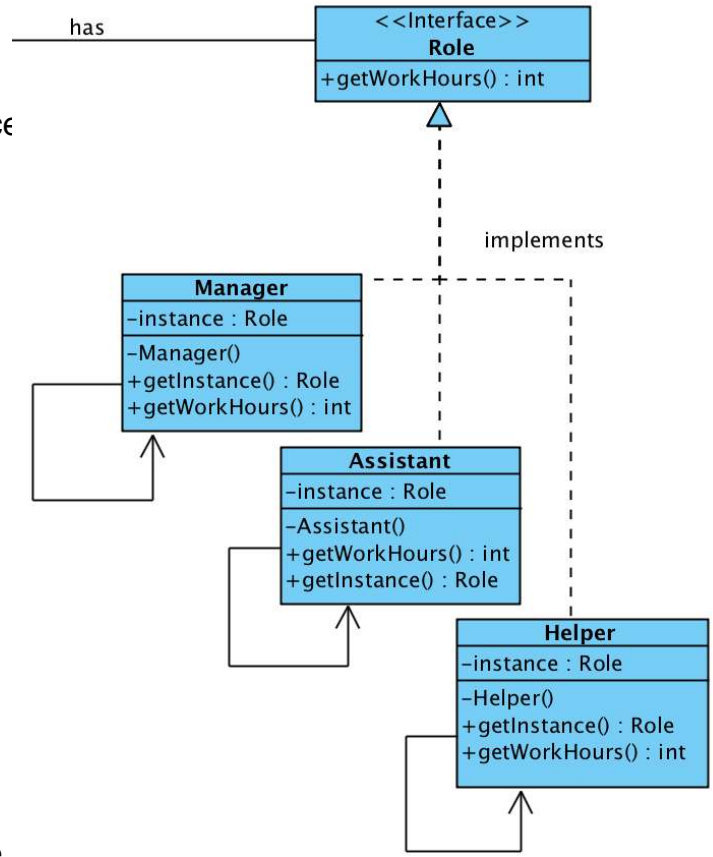
Singleton Pattern. (3 Marks)

The situation can be addressed to ensure that there will be ever a single object instance per state to be created at run-time.

Applying Singleton Pattern will enforce this Requirement.

- instance : Role (1 Mark)
- Manager () constructor (1 Mark)
- getInstance() : Role (1 Mark)

3 Marks Max.




*4 Marks total for the following:*

```
public class Manager implements Role_Type
{
   private static final Role_Type instance = new Manager ();
   // 1 Mark

   private Manager () {}
   // 1 Mark

   public static Role_Type getInstance () {
      return instance;
   }
   // 1 Mark

   public int getWorkHours() {
      return 20;
   }
   // 1 Mark
}
```

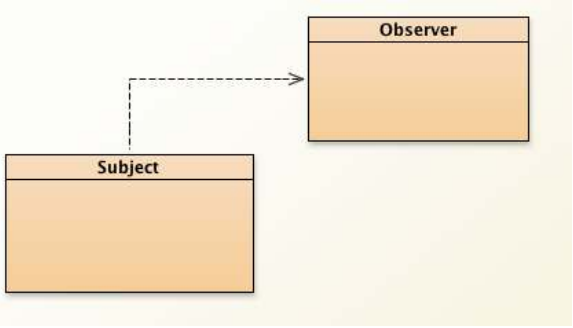
5(c)

It can be accomplished using **Observer Pattern (2 Marks)**: Person being the Observer and Event being the Subject.

Person can subscribe to the subscribers list (can also unsubscribe), notify_all() function will broadcast the update message to all the subscribers. **(2 Mark for details)**

```
import java.util.*;

public class Event
{
   private String event_Name;
   private int Date;
   private String Venue;

   private ArrayList<Person> subscribers = new
ArrayList<Person>();
   private String newsletter;

   public Event(String _name, int _Date, String _venue)
   {
      event_Name  = _name;
      Date = _Date;
      Venue = _venue;
   }

   public void subscribe (Person p) {
     p.update(this);
     subscribers.add(p);
   }

   public void unsubscribe (Person p) {
      subscribers.remove(p);
   }

   public void notify_all () {
      for (Person o: subscribers)
         o.update(this);
   }

   public void setNewsLetter(String news) {
      newsletter = news;
   }

   public String getNewsLetter() {
      return newsletter;
   }

}
```

```
class Person
{
   private String name;
   private int yearOfBirth;
   private Address address;

   Person(String name, int yearOfBirth)
   {
      this.name = name;
      this.yearOfBirth = yearOfBirth;
   }

   public void setAddress(String street, String town,
String postCode)
   {
      address = new Address(street, town, postCode,
"");
   }

   public Address getAddress()
   {
      return address;
   }

   public void update(Event e) {
      System.out.println (e.getNewsLetter());
   }

}
```




**4 Marks for correct coding example / code fragments.**

## 6. Professional Ethics in Software Engineering (5 Marks) – CILO 5

***Pico* violates (2 Marks):**

Pico violates at least three codes of ethics: (2 %)

1) [*Client and employer*] Act in the best interests of their clients and employer
2) [*Product*] Develop and maintain the product (e.g., software and documentation) with the highest standards possible
3) [*Judgment*] Maintain integrity and independence (of oneself)

**1 marks for each point. 2 marks max.**

***Mac* follows (3 Marks):**

Mac follows at least five codes of ethics: (3%)

1) [*Client and employer*] Act in the best interests of their clients and employer
2) [*Product*] Develop and maintain the product (e.g., software and documentation) with the highest standards possible
3) [*Management*] Promote an ethical (e.g., equal opportunity, match task against skill level instead of friendship) approach in management of subordinates (who are managed by you)
4) [*Profession*] Advance the integrity and reputation of the profession as software engineers
5) [*Colleagues*] Be fair and supportive to colleagues

**1 mark for each point. 3 marks max.**