SIT232 Object-Oriented Development Exam Solution – Trimester 1, 2015

This solution covers Section B (short answer) of the examination paper only. Section A (multiple choice) is drawn from a large pool of questions which are marked automatically by Scanning within DSA. To maintain the value of the pool of multiple choice questions, no reproduction of those questions or answers is made available to students.

Note that all answers provided are only for background, it is not expected that a student will have the exact same answer, or necessarily the same level of detail.

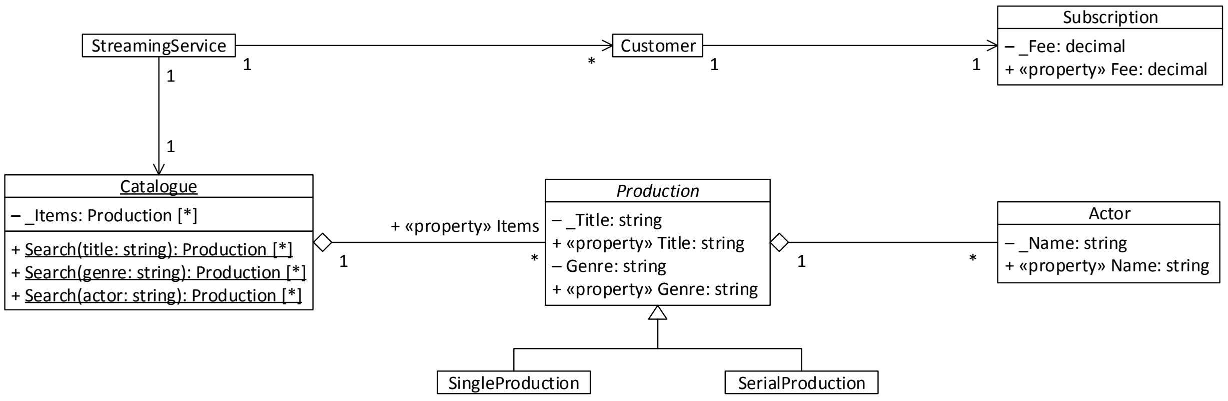
Question a.

a) Consider the following problem statement:

> *You have been hired by a newly established TV/video streaming service to develop a new software application to track the titles they offer and what content is being viewed by their customers. Customers pay a monthly subscription fee to gain access to standard titles, and can optionally pay an increased fee for access to premium titles. Titles are then divided into single productions, e.g., movies, and serial productions, e.g., television series, which consist of a set of two or more single productions. The different titles offered must be searchable by title (the name of the show), genre (action, comedy mystery, etc.), and/or actor/actress. This search function must be able to be performed both by customers wishing to view titles and by the company wishing to identify which customers have viewed which titles.*

Your task is to prepare a UML class diagram illustrating the concepts in the problem statement above. Your class diagram must demonstrate the following elements:

- Attributes, properties, and methods;
- Collections;
- Abstract classes;
- Inheritance;
- Associations (any type); and
- Cardinality.



StreamingService
Customer
Subscription
– _Fee: decimal
+ «property» Fee: decimal
Catalogue
– _Items: Production [*]
+ Search(title: string): Production [*]
+ Search(genre: string): Production [*]
+ Search(actor: string): Production [*]
+ «property» Items
Production
– _Title: string
+ «property» Title: string
– Genre: string
+ «property» Genre: string
Actor
– _Name: string
+ «property» Name: string
SingleProduction
SerialProduction
1
*

Note that the above UML is only one example solution, many others will be acceptable.
Marking scheme (15 marks):

7.0 marks (1.0 mark each) – the following elements are applied appropriately in the UML:
attributes, properties, methods, collections/cardinality, abstract classes, inheritance, and associations.

6.0 marks (2.0 marks each) – the following elements are represented appropriately in the UML:
productions (including types of productions), customers and subscriptions, and the ability to search the catalogue (including by title, by genre, and by actor/actress).

2.0 marks – the notation in the UML is otherwise correct

Question b.

b) For this task you are required to write C# code satisfying the following requirements:

- An abstract class named Result which has:
  - An string attribute for the student name;
  - An integer attribute for the mark achieved;
  - Read-only properties encapsulating the above attributes;
  - A method ScaleUp which allows only derived classes to provide a new higher value for the mark;
  - A custom constructor which takes name and mark parameters and initialises the attributes defined above; and
  - A ToString() method which returns a string containing the asset description and value in the format:

    name: *mark*
- An class named ScaledResult which inherits from the Result class and has:
  - An integer attribute for the rate;
  - A read-only property encapsulating the above attribute;
  - A custom constructor which takes name, mark, and rate parameters, passes the name and mark to the base class constructor and initialises the rate attribute defined above;
  - A Scale() method, which can be invoked by any other class, which increases the mark of the student according to the following formula:

$$new\_mark = old\_mark + (old\_mark * \frac{rate}{100})$$

```
abstract class Result
{
    private string _Name;
    public string Name
    {
        get { return _Name; }
    }

    private int _Mark;
    public int Mark
    {
        get { return _Mark; }
    }

    protected void ScaleUp(int newMark)
    {
        _Mark = newMark;
    }

    public Result(string name, int mark)
    {
        _Name = name;
        _Mark = mark;
    }

    public override string ToString()
    {
        return string.Format("{0}: {1}", _Name, _Mark);
    }
}
```

```
class ScaledResult : Result
{
    private int _Rate;
    public int Rate
    {
        get { return _Rate; }
    }

    public ScaledResult(string name, int mark, int rate)
        : base(name, mark)
    {
        _Rate = rate;
    }

    public void Scale()
    {
        double newMark;
        newMark = Mark;
        newMark += newMark * _Rate / 100;
        ScaleUp((int)newMark);
    }
```

Marking scheme (15 marks):
Result class:

1.0 mark – class is declared with abstract keyword
2.0 marks – name and mark attributes and read-only properties correct
2.0 marks – constructor accepts parameters and initialises attributes correctly
2.0 marks – ScaleUp method has protected accessibility and is otherwise correct
2.0 marks – ToString returns correctly formatted string and is otherwise correct

ScaledResult class:

1.0 mark – class inherits from Result class
1.0 mark – rate attribute and read-only property correct
2.0 marks – constructor accepts and initialises rate and passes remaining parameters to base
1.0 mark – Scale method correctly applies formula and saves result using ScaleUp method
1.0 mark – Scale method correctly applies non-integer data type for calculation

Penalise up to 2.0 marks is there are clear problems with the syntax.
