FINAL EXAMINATION

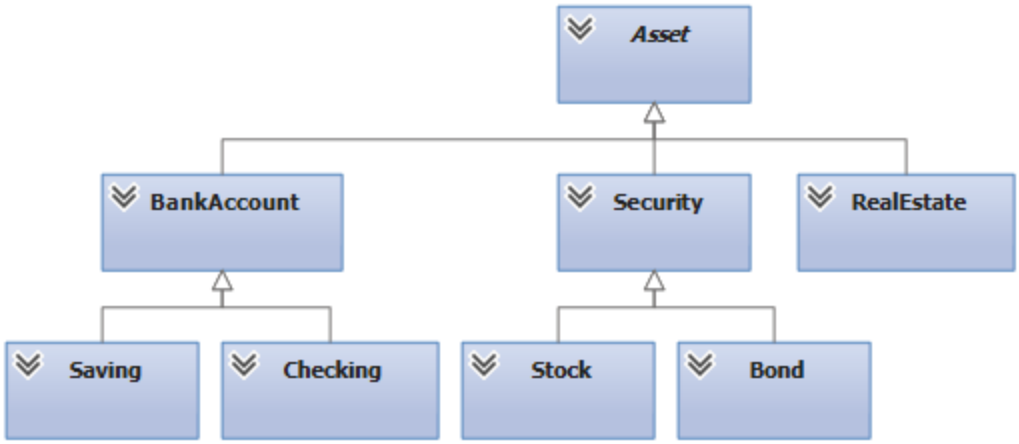
Course: **CS3373 - Advanced object oriented programming for windowing environments**	
Time: 120 minutes	Term: 1 – Academic year: 2016-2017
Lecturer(s): Assoc. Prof. Tran Minh Triet	
Student name:	Student ID:

(Notes: Neither books nor laptops, phones allowed)

<Followings are the questions and/or requirements>

Question 1: Polymorphism **(2.5 marks)**

You are given a class diagram as follows:
Note: ***`Asset`*** is an abstract class!



Which of the following code fragments are ***correct***? Briefly ***explain*** your choice.

Code fragment	Correct or not?	(Brief) Explanation
`BankAccount pAsset;` `pAsset = new Stock;`	☐ Correct ☐ Incorrect	…… …… …… ……
`Stock pAsset;` `pAsset = new Security;`	☐ Correct ☐ Incorrect	…… …… …… ……

Code fragment	Correct or not?	(Brief) Explanation
`Asset pAsset;` `pAsset = new Asset;`	☐ Correct ☐ Incorrect	…… …… …… ……
`Bond pAsset;` `pAsset = new Bond;`	☐ Correct ☐ Incorrect	…… …… …… ……

`BankAccount pAsset;`
`pAsset = new Saving;`
`Asset p;`
`p = (Asset) pAsset;`
	☐ Correct
	☐ Incorrect
		……
		……
		……
		……

`BankAccount pAsset;`
`pAsset = new Saving;`
`BankAccount p;`
`p = (BankAccount) pAsset;`
	☐ Correct
	☐ Incorrect
		……
		……
		……
		……

`BankAccount pAsset;`
`pAsset = new Saving;`
`Checking p;`
`p = (Checking) pAsset;`
	☐ Correct
	☐ Incorrect
		……
		……
		……
		……

`BankAccount pAsset;`
`pAsset = new Saving;`
`Saving p;`
`p = (Saving) pAsset;`
	☐ Correct
	☐ Incorrect
		……
		……
		……
		……

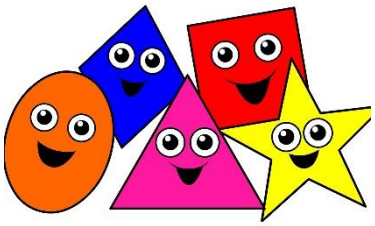
Code fragment	Correct or not?	(Brief) Explanation

`BankAccount pAsset;`
`pAsset = new Saving;`
`pAsset = new BankAccount;`
`Saving p;`
`p = (Saving) pAsset;`
	☐ Correct
	☐ Incorrect
		……
		……
		……
		……

`BankAccount pAsset;`
`pAsset = new Saving;`
`Asset p;`
`p = (BankAccount) pAsset;`
	☐ Correct
	☐ Incorrect
		……
		……
		……
		……

Question 2: **(2.0 marks)**

Class *Shape* is the abstraction of all shapes. Class Triangle and Circle inherit from the abstract class *Shape*. The function *Clone()* in class *Shape* is used to generate a similar object of a given shape object. The function *GetName()* in class *Shape* returns the name of that shape.



```
abstract class Shape {
    public abstract Shape Clone();
    public abstract string GetName();
}
class Point {
    public double X;
    public double Y;
}
```

```
class Triangle: Shape {
    public Point A;
    public Point B;
    public Point C;
}
class Circle: Shape {
    public Point Center;
    public double Radius;
}
```

Class ShapeManager is used to manage the creation of all shape objects.

```
class ShapeManager {
   protected List<Shape> SampleShapes; // samples of all shapes
   public ShapeManager() {
   // init samples of all shapes…
   }
   public Shape CreateShape(string strShapeName)  {
   // create a shape object corresponding to the name strShapeName
   }
}
```

a) Implement the ***default constructor*** of class `ShapeManager` to ***initialize all sample shapes***.

(0.5 mark)

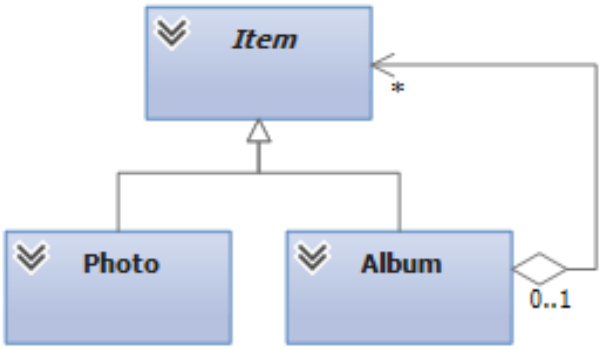
b) Implement the function *`Clone()`* for class `Circle`. *(0.5 mark)*

c) Implement the function `GetName()` for class `Rectangle`. *(0.5 mark)*

d) Implement the function `CreateShape()` of class `ShapeManager` ***to create a shape with the name*** `strShapeName`. *(0.5 mark)*

Question 3: **(4.0 marks)**

You have a lot of photos and you want to write a program to manage them. An ***album*** can contain ***multiple photos and sub-albums***. A ***photo*** or an ***album*** is called an ***item***. Each item (photo or album) has its own name (`string`), date of creation (`DateTime`), tags (`string`), and rating (`double`, from 0 to 5 stars). A photo has its size (in bytes) and the pathname (path and filename) of the corresponding file.



a) ***Define*** the ***attributes*** of the following classes: `Item`, `Photo`, and `Album`. *(1.0 mark)*

Note: `Item` is an abstract class. Do NOT define any methods/functions in these classes.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

b) ***Define*** and ***implement methods*** of your defined class(es) to ***input data of an item*** (*1.0 mark*)

Hint: - For a photo, you simply ask a user to input the attributes of that photo.

- For a folder, you should allow a user to input the attributes of that album and the information of all photos and sub-albums in that album ***recursively***.

c) You want to search for all photos that satisfy an ***arbitrary criterion*** in a given album.

In the abstract class `Criterion`, the abstract method `IsSelected()` checks if an item (a photo or an album) satisfies a given condition.

```
class abstract Criterion {
public abstract bool IsSelected(Item item);
}
```

Define and ***implement*** the two classes `SelectByName` and `SelectByRating` inheriting from `Criterion` to select an item by its ***name*** or ***minimum rating.*** *(1.0 mark)*

Hint: Define and implement the constructor; then override the method `IsSelected()` in each class (`SelectByName` and `SelectByRating`).

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

d) The abstract method *Search()* in class *Item* returns the list of all items that satisfy condition.

```
class abstract Item {
public abstract List<Item> Search (Criterion condition);
}
```

You should ***override this method*** in the two classes `Photo` and `Album`. *(1.0 mark)*

Question 4: **(1.5 marks)**

a) **Numbers** *(0.5 mark)*

There are different types of numbers, such as integers, floa ting-point numbers, fractions, etc.

Class *Number* is the abstraction of a number. In this class, the abstrac function *GetValue*() returns the value (as a floating-point number) of the number.

```
abstract class Number {
// return the value as a floating-point number
public abstract double GetValue();
}
```

```
class MyInteger: Number {
protected int v;
public MyInteger(int v) {
   this.v = v;
}
public override double GetValue() {
   return (double)this.v;
}
}
```

Define and ***implement*** class `MyFraction` inheriting from *Number* to represent a fraction.

Hint: you should define the attributes, constructor, and function *GetValue*() of class `MyFraction`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

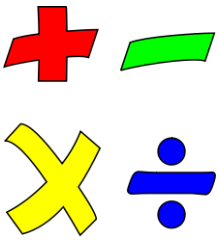
.....

.....

.....

.....

b) **Arithmetic Operators** *(0.5 mark)*



There are different arithmetic operators: add, subtract, multiply, and divide. To perform different arithmetic operations on two numbers, we define the abstract class *Operator* to represent an operator. In this class, the abstract function *Evaluate*() returns the result (as a floating-point number) of the operation on two given numbers (x and y).

```
abstract class Operator {
public abstract double Evaluate(Number x, Number y);
}
```

Define and ***implement*** class AddOperation inheriting from *Operator* to add two numbers.

Hint: you should use the function *GetValue*() of a number.

c) **Arithmetic Expressions** *(0.5 mark)*

We can perform different arithmetic operations on t wo numbers (of different types, such as integers, floating-point numbers, of fractions). In class `Expression`, the attributes `x` and `y` can be of any types inheriting from *`Number`*. The attribute `op` represents the operator to be performed on `x` and `y`.

```
class Expression {
protected Number x;
protected Number y;
protected Operator op;
public double Evaluate() {...}
}
```

Implement function `Evaluate()` of class `Expression` to return the result of the arithmetic operation on `x` and `y`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

GOOD LUCK TO YOU ☺