NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

MIDTERM TEST FOR CS1020

Semester 2: 2010/11

CS1020 – Data Structures and Algorithms I

March 2011 Time allowed: 1 hour 30 minutes

INSTRUCTIONS TO CANDIDATES

1. This test paper consists of **FIFTEEN (15)** questions and comprises **ELEVEN (11)** printed pages.

2. This is a **CLOSE BOOK** test. You are allowed to bring in ONE (1) piece of A4 handwritten reference sheet.

3. Answer all questions. Use the provided OCR form for MCQ questions (Q1 to Q12). Use the provided answer sheet for short questions (Q13 to Q15).

4. Shade and write down your matriculation number on the OCR form. Fill in your Matriculation Number with a pen, clearly on every page of your ANSWER SHEETS.

5. You may use pencil to write your codes. Pen is preferred for other questions.

6. You must submit both the OCR form and the ANSWER Sheet. It is your responsibility to ensure that you have submitted both to the invigilator at the end of the test.

SECTION A (12 Multiple Choice Questions: 48 Marks)

Each question has one correct answer. Write your answer in the space provided on the ANSWER SHEETS. 4 marks are awarded for each correct answer and no penalty for wrong answer.

1. Given an ADT specification, the different implementations **must meet** which of the following requirement(s)?

   i. All operations in the specification must be supported with the same parameters and return type
   ii. All operations in the specification must have the same time and space efficiency
   iii. Individual implementation must have different object attributes

   a. (i) only.
   b. (i) and (ii) only.
   c. (ii) and (iii) only.
   d. (i) and (iii) only.
   e. (i), (ii) and (iii).

2. Which of the following statement(s) regarding **instance method** is/are **TRUE**?

   i. Instance method must be invoked with an object reference
   ii. Instance method can be inherited
   iii. Instance method cannot be declared with private visibility

   a. (i) only.
   b. (i) and (ii) only.
   c. (ii) and (iii) only.
   d. (i) and (iii) only.
   e. (i), (ii) and (iii).

3. Which of the following statement regarding **Object Oriented Concepts** is **TRUE?**

   a. Objects of a single class can have different operations (instance methods) defined for them.
   b. Objects of the same class share one copy of the object attributes (instance variables) between them.
   c. An object oriented program must have everything coded in a single class.
   d. Objects of different classes can access each other private attributes due to encapsulation principle.
   e. None of the above.

4. Given two classes as follows:

```
class A
{
    private int A_i;

    //other code not shown
    public void print() {
         System.out.println( A_i );
    }
}

class B extends A
{
    private int B_i;

    //other code not shown
    public void print() {
        //Implementation
    }
}
```

Which of the following implementation(s) for class B's print() method correctly print out the value of object attribute **A_i** followed by **B_i**?

i.
```
super.print();
System.out.println( B_i );
```
ii.
```
System.out.println( A_i );
System.out.println( B_i );
```
iii.
```
super.print();
this.print();
```

a. (i) only.
b. (i) and (ii) only.
c. (ii) and (iii) only.
d. (i) and (iii) only.
e. (i), (ii) and (iii).

5. Which of the following statement is TRUE regarding encapsulation principle?

a. The main purpose of encapsulation is to increase efficiency of implementation.
b. Encapsulation allows the user of a class to have complete access to the internal implementation details of a class.
c. Having private methods in a class violates the encapsulation principle.
d. To correctly apply encapsulation principle, we must always provide a full set of accessor methods and mutators method for all private attributes.
e. None of the above.

6. Given the following code snippet on Java exception handling.

```
try {
      f();                    //f() can throw a E1 exception
      System.out.println( "after f()!" );
      g();                    //g() can throw a E2 exception
      System.out.println( "after g()!" );
} catch ( E1 expObj ) {
      System.out.println( "E1 exception caught!" );
} catch ( E2 expObj ) {
      System.out.println( "E2 exception caught!" );
}
```

Which of the following output is **impossible** to get when running the code above?

a.
```
E1 exception caught!
```

b.
```
after f()!
after g()!
```

c.
```
after f()!
E2 exception caught!
```

d.
```
after f()!
after g()!
E2 exception caught!
```

e. None of the above

7. Which of the following statement(s) is/are TRUE regarding the exception mechanism?

   i. It involves both the coder of a method and the user of a method.
   ii. By raising exception, the coder of a method can handle all exceptional scenarios within the method itself.
   iii. The user of a method can never be sure what are the exceptions thrown by a method.

a. (i) only
b. (ii) only
c. (i) and (ii)
d. (i) and (iii)
e. (i), (ii) and (iii)

8. Given the generic Pair class from lecture 4:

```
class Pair <S,T> {
      private S _first;
      private T _second;

      public Pair( S a, T b ) {
            _first = a;
            _second = b;
      }
      public S getFirst() { return _first; }
      public T getSecond() { return _second; }
}
```

Which of the code fragment(s) below can be compiled with no error?

i.
```
Pair<Double, String> p =
        new Pair<Double,String> (3.14, "Pi");
String s = p.getFirst() + " is not accurate.";
```

ii.
```
Pair<String, Integer> p =
        new Pair<String, Integer> ("Value", 1458);
int i = p.getSecond();
```

iii.
```
Pair<Integer, Double> p =
        new Pair<Integer, Double> (1, 2.02);
int k = p.getFirst() + p.getSecond();
```

a. (i) only.
b. (i) and (ii) only.
c. (ii) and (iii) only.
d. (i) and (iii) only.
e. (i), (ii) and (iii).

9. Suppose we implement a **Person** class using the same generic Pair<S,T> class from MCQ 8. Here's the partial implementation:

```
class Person {

    //attempt to use a pair to represent the name and occupation of a person
    private Pair<String, String> nameJob;

    //attempt to use a pair to represent the address and phone no of a person
    private Pair<String, Integer> addressPhone;

   //other code not shown
}
```

Which of the following comment is **TRUE** regarding the object attribute declaration above?

a. The **`Person`** class cannot be compiled since any generic class can be initialized to the given user data types once per program only. So, the declaration of **`addressPhone`** conflicts with **`nameJob`**.
b. The **`Person`** class must be a generic class itself in order to use the generic **`Pair`** class.
c. The generic attributes (object variables) must be initialized at the point of declaration. So, the above code failed to compile.
d. The generic attributes (object variables) must have public visibility.
e. None of the above.

10. Given the following method:

```
public static String mystery( String old )
{
    String temp = old;

    for (char c = 'a'; c <= 'z'; c++) {
        temp = temp.replace( c, (char)(c - ('a' - 'A')) );
    }

    return temp;
}
```

`replace`**`( char oldChar, char newChar )`** is a predefined method for String class that replaces all occurrences of the character **`oldChar`** in the string with character **`newChar`**. You can assume that ('a' – 'A') evaluates to a positive integer.

Which of the following statement(s) is/are **TRUE** regarding this method?

i. **`mystery( "abcxyz" )`** returns the string **`"ABCXYZ"`** as result

ii. **`mystery( "abcabc" )`** returns the string **`"ABCabc"`** as result

iii. **`mystery( str )`** will give well defined result only if the **`str`** contains only letters (uppercase and lowercase).

a. (i) only.
b. (i) and (ii) only.
c. (ii) and (iii) only.
d. (i) and (iii) only.
e. (i), (ii) and (iii).

11. Below is an attempt to enlarge an existing array:

```
public static void enlargeArray( int[] iA,
                                 int extraCapacity )
{
      int[] temp;

      temp = new int[ iA.length + extraCapacity];
      for (int k = 0; k < iA.length; k++) {
            temp[k] = iA[k];
      }
      iA = temp;
}
```

If we use the above method as follows:

```
int[] intArray = { 1, 2, 3, 4, 5 };

enlargeArray( intArray, 5 );
```

Which of the following statement is TRUE after the above code is executed?

a. **`intArray`** is now size 10, with the first 5 elements as **`{1, 2, 3, 4, 5}`**
b. **`intArray`** is now size 10, but all 10 elements are not defined
c. **`intArray`** is still size 5, with all 5 elements retained as **`{1, 2, 3, 4, 5}`**
d. **`intArray`** is still size 5, with all 5 elements undefined.
e. None of the above.

12. Given two classes as follows:

```
class X
{
  protected int xInt;

  public X() {
        xInt = 123;
  }
  public X(int m) {
        xInt = m;
  }
}

class Y extends X
{
  protected int yInt;
  public Y(int m) {
        super();
        yInt = m;
  }
  public void print() {
        System.out.println(xInt + yInt);
   }
}
```

Which of the following code fragment(s) and their associated behavior is **correct**?

i.
```
Y objY = new Y(4000);
objY.print();
```

Code Behavior: Result `"4123"` printed

ii.
```
Y objY = new Y(0);
objY.print();
```

Code Behavior: Result `"0"` printed

iii.
```
Y objY = new Y();
objY.print();
```

Code Behavior: Compilation error

a. (i) only.
b. (i) and (ii) only.
c. (ii) and (iii) only.
d. (i) and (iii) only.
e. (i), (ii) and (iii).

SECTION B (3 Short Questions: 52 Marks)

Question 13 (16 marks)

One simple idea to compress a string of characters is to group adjacent identical characters into a unit. We can then specify each unit by two pieces of information: ***the character*** and ***number of occurrences***. This results in a shorter description of the original string if there are many duplicated characters.

For example:

Original string (Adjacent identical characters are shown in the same font)	**After compression**
AAAAABBB	**('A',5), ('B',3)** meaning: 5 copies of 'A' followed by 3 copies of 'B'
AAAAAB**CCCC**AA	**('A',5), ('B',1), ('C',4), ('A',2)**
A**B**C**D**	**('A',1), ('B',1), ('C',1), ('D',1)**

Given the **`Pair<S, T>`** generic class and **vector**, implement a simple **static method** to perform the compression. The method takes in a string and performed the grouping as described. Each unit is then stored as a pair of a character and an integer. The pairs are then stored in a vector, with the 1st pair describing the first unit in the original string.

For example, if we call

```
result = compressString( "AAAAABBB" );
```

`result` should be a vector that contains two **`Pair`** references: **`{ ('A', 5), ('B', 3)}`**.

Give your Java implementation in the given answer sheet.

Question 14 (20 marks)

Every year, the student admission office in NUS needs to process thousands of admission applications. In this question, we will attempt to model a simplified version of this process as an object oriented program.

Suppose there are 3 types of applicants only: the A-Level students, the polytechnic diploma holder and the international students. All applicants have a name as well as an unique application number (an integer). Each type of applicant has their own qualification metric:

1. A-Level student: aggregate score of their A-Level paper, an integer.
2. Polytechnic diploma holder: the GPA (Grade point average) at the end of their study, a double value.
3. International student: An aggregate score of their final year papers, an integer.

Since the qualification metric is different for each category, the admission office devised a uniform scale, the **university admission point** (an integer value), to quantify the quality of every applicant. Essentially, each type of applicant will convert their own qualification metric based on certain formula to the **u**niversity **a**dmission **p**oint (**UAP**).

Part A) Based on the description above, draw an UML class diagram to represent the important entities in the scenario. **You only need to model the 3 types of applicant, together with the necessary operations and object attributes.** For simplicity, you only need to ensure the following operations are available for each type of applicant:

- ***`getName()`*** : Accessor method for the name of applicant
- ***`getAID()`***: Accessor method for the application number of the applicant
- ***`toUAP()`***: Convert the qualification metric of the applicant to the University Admission Point (the exact formula is not important in this question)

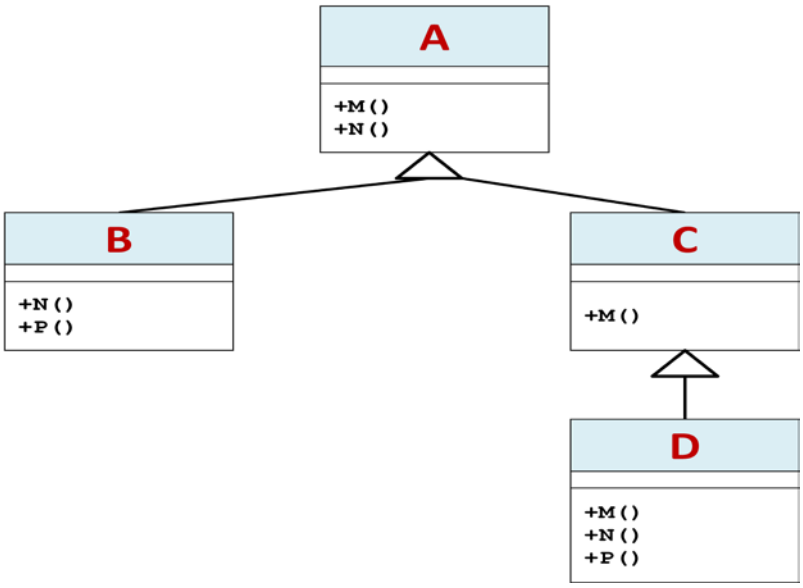
Your design is judged by the correct use of class relationship and method overriding. The classes should make processing all applicants uniformly possible and simple. In the answer sheet, the three class icon for each type of applicant is already provided, you are free to introduce other class if necessary. (**10 marks**)

Part B) Based on your design in part (a), implement a **static method** that:

- Takes in **an array of applicants**. The applicant can be of any type.
- Return the application number of the best applicant in this array. Best applicant is the applicant with the **highest UAP** in this array. (**10 marks**)

Question 15 (16 marks)

Given the following class hierarchy in UML notation:



For simplicity, you can assume that all methods in the various classes simply prints out message of the form **`<class_name>.<method name>`**, e.g. the method ***`M()`*** in class A prints out "**`A.M`**". Note also that if a class overrides an inherited method, the method's name will appear in the class icon, e.g. class **`B`** overrides the method **`N()`**. Otherwise, the inherited method remains **unchanged** in the subclass.

For each of the code fragments below, indicate whether:

- The code will cause compilation error. Briefly explain the reason. **OR**
- The code can compile and execute. Supply the execution result in this case.

See below for two examples.

Code Fragment (Example)	**Compilation Error**	**Execution Result**
`A a = new A();` `a.M();`		A.M
`A a = new A();` `a.K()`	method K() not defined in class A	

Note that any code fragment with both choices filled will be taken as incorrect. Each code fragment should be considered on its own, independent from other code fragments. Use answer sheet for the rest of the code fragments.

~~~ END OF PAPER ~~~