ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

CE/CZ2002

$2.80

NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 1 EXAMINATION 2014-2015

CE2002/CZ2002 – OBJECT ORIENTED DESIGN & PROGRAMMING

Nov/Dec 2014 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 4 questions and comprises 9 pages.
2. Answer ALL questions.
3. This is a closed-book examination.
4. All questions carry equal marks.
5. APPENDIX A shows Java code listing referenced by Question 1.
6. APPENDIX B shows the API specification for the Java class `ArrayList`.
7. APPENDIX C shows the Sequence Diagram referenced by Question 3.

---

1. (a) Give an example of defining a *static* variable in Java using correct syntax and explain TWO differences between *static* variable and *instance* variable.

(6 marks)

(b) List TWO differences between *static* binding and *dynamic* binding.

(4 marks)

(c) Study the Java code listing in APPENDIX A (page 7).

(i) State the line number(s) of the incorrect code, explain the reason, and write the correct code.

(8 marks)

Note: Question No. 1 continues on Page 2

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

(ii) After correcting the code, draw the relationship between `AnAbstract`, `IInterface`, `Concrete1`, and `Concrete2` in a Class Diagram. Show all the methods in each class or interface.

(7 marks)

2. You are required to write a Java application program to print out the loan records of a material in a library. Each record contains the information about the name of the person who borrowed the material and how many days the person borrowed the material. A material can be a book or a magazine. For a book, the program only prints out the records where the book was borrowed for more than a WEEK. For a magazine, the program prints out the records where the magazine was borrowed for more than THREE days.

(a) Write the code for the `Record` class that has the instance variables of `nameOfBorrower` and `numOfDays`, a constructor, and the instance methods of `getDays` and `toString`.

- `nameOfBorrower`: name of the person who borrowed the material.
- `numOfDays`: number of days for which the person borrowed the material.
- the constructor initializes the values of all instance variables.
- `getDays`: accessor method to get the `numOfDays` value.
- `toString`: method to return the name of the person who borrowed the material and the number of days for which the person borrowed the material.

(5 marks)

(b) Write the code for the abstract class `Material` that has the instance variables of `id` and `records`, a constructor, the instance methods of `addRecord` and `getRecords`, and an abstract method `genReport`.

- `id`: the identity of the material.
- `records`: the list of all loan records of the material. Store the list of loan records in an `ArrayList`.
- the constructor initializes the values of all instance variables.
- `addRecord`: method to add a loan record to the list of loan records.
- `getRecords`: method to return the list of all loan records.
- `genReport`: an abstract method.

Note: The `ArrayList` class belongs to the java.util package and its API specification is given in Appendix B (page 8).

(5 marks)

Note: Question No. 2 continues on Page 3

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

(c) Two subclasses, `Book` and `Magazine`, are derived from the `Material` class. Each subclass has a constructor to initialize the values of the inherited instance variables. Each subclass also has a `genReport` method and provides implementation for the method. The `genReport` method returns the list of loan records for a material (a book or a magazine). Write the code for the two subclasses.

(7 marks)

(d) Write an application class `PrintReport` to have two `printReport` methods. One `printReport` method prints out the loan records of a book, and another `printReport` method prints out the loan records of a magazine. Also write the `main` function in the application class to demonstrate **static binding** of the `printReport` methods for a book and a magazine.

To demonstrate **dynamic binding** of the `printReport` method, write another version of the application class `PrintReport` to have only one `printReport` method in the application class, regardless of which `Material` subclass is passed as the argument of the `printReport` method.

(8 marks)

3. (a) Program Listing Q3a and Q3b shows the code listing of the declaration of `Student` class and `Employee` class respectively written in C++ language.

```
// in student.h header file
#include <string>

using namespace std;
class Student
{
  private:
    string name;
    string matric;

  public:
    Student(string name, string matric);
    string getMatric() ;
    void print() ; // print the student name
};
```

Program Listing Q3a

Note: Question No. 3 continues on Page 4

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
// in employee.h header file
#include <string>

using namespace std;

class Employee
{
  private:
    string  employer;
    double  wage; // full monthly wage

  public:
    Employee(string employer, double wage) ;
    virtual double getWage() ;
    virtual void print() ; // print the employee name
};
```

Program Listing Q3b

(i) Using inheritance, we want to derive a `PartTimeStudent` class from both `Student` class and `Employee` class. This class has an attribute `proRate` (a double) that indicates the prorated proportion per month. The default prorated proportion per month is 0.5 (half of full monthly wage). The class constructor takes FIVE parameters; `employer`, `wage`, `name`, `matric` (corresponding to the member attributes of its base classes) and its own member attribute, `proRate`.

The class also has TWO methods; `getWage()` and `print()`. The `getWage()` function returns the prorated wage as *prorate multiplied by the monthly wage.* The `print()` function uses the `print()` functions from its base classes and prints information to the console in the following format :

```
Employer :  <employer name>
Student name: <student name>
ProRate Wage:  <result of getWage() >
```

Write the full declaration and implementation code for `PartTimeStudent` class. Assume the implementations of both `Student` class and `Employee` class are already done.

(10 marks)

(ii) By using *pointer (*) AND ampersand (&)*, write a `main` function to demonstrate the use of `print` function to show its polymorphic behavior.

(3 marks)

Note: Question No. 3 continues on Page 5

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

(b) The **UML Sequence Diagram** in Appendix C (page 9) shows the objects' interaction of a scenario flow in a particular application. Using the details depicted in the diagram, write the preliminary JAVA code and methods for the class `ClassA`. You may make appropriate assumptions on the method parameters, return types and return value(s) if they are not stated in the diagram.

(12 marks)

4. (a) State the THREE symptoms of rotting design taught in the course and explain ONE of the symptoms in the context of software design.

(5 marks)

(b) Study the following description of an eAnnouncement application:

An eAnnouncement application is to be developed to make announcement notification to all registered recipients. The recipient can be an individual or one who joined in a group. During registration, the recipients must provide their user identity, name, email address and optionally, mobile number and other instant messaging accounts such as WhatsApp, MSN messenger. Recipients can select their preferred mode of notification, for example, email (default), Short Message Service (SMS), instant message, etc.

An eAnnouncement can be created as an HTML text or plain text. HTML text allows better visual presentation via fonts, table or images, etc. Besides the main text, the announcement will have a subject title, date of notification and a selection of the recipients or the targeted group of recipients. Depending on the mode of notification selected by recipient, the announcement will be presented in a way suitable for the type of mode.

(i) You are being tasked to identify the entity classes needed to build the eAnnouncement application based on the description above.

Show your design in a Class Diagram. Your Class Diagram should show clearly the relationship between classes, relevant attributes (at least TWO) and, multiplicities, association names, if any. You need not show the class methods.

(14 marks)

Note: Question No. 4 continues on Page 6

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

(ii) State the Open-Closed design principle and use it to explain your design to cater to the different <u>*modes of notification*</u> supported by the eAnnouncement application. [*Hint : You can assume there is a method called* `sent` *and use it in your explanation*].

(6 marks)

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

APPENDIX A : Java Code Listing

```
public abstract class AnAbstract{
   public int doWork() {...}
   public final void reviseWork() {...}
   public abstract void showWork();
}

public interface IInterface{
   public void doWork();
}

public final class Concrete1 extends AnAbstract
                                  implements IInterface {
   public void reviseWork() {...}
   public String showWork() {...}
}

public class Concrete2 extends Concrete1{
   public void showWork() {...}
}
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

APPENDIX B : ArrayList API Specification

java.util

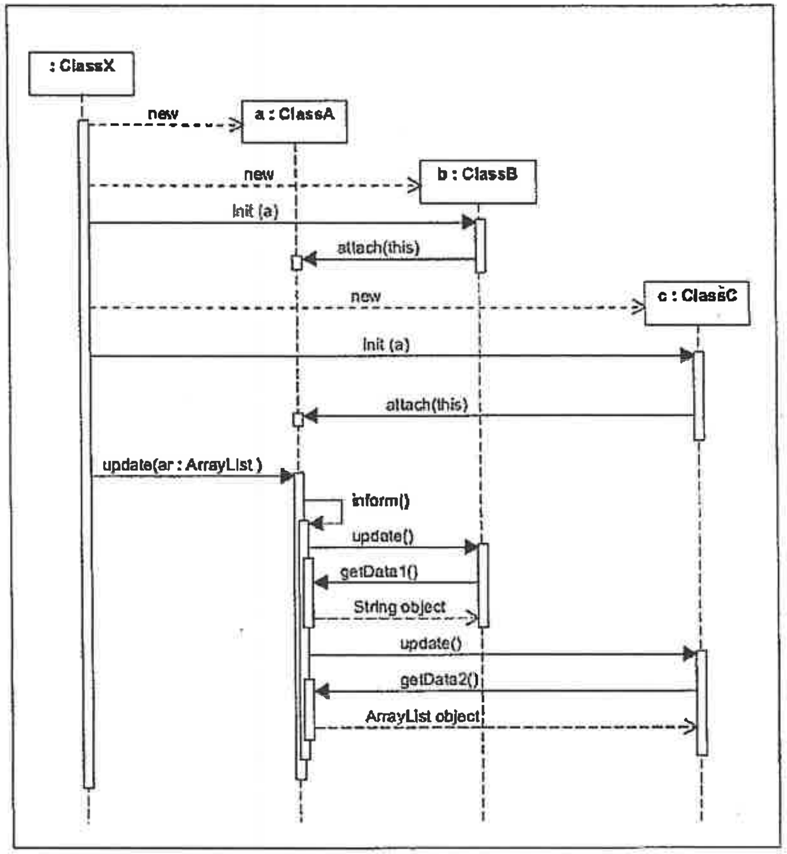
Class ArrayList<E>

ArrayList Methods

Modifier and Type	Method and Description
`boolean`	`add(E e)` Appends the specified element to the end of this list. Returns `true` if this collection changed as a result of the call. (Returns `false` if this collection does not permit duplicates and already contains the specified element.)
`void`	`add(int index, E element)` Inserts the specified element at the specified position in this list.
`void`	`clear()` Removes all of the elements from this list.
`Object`	`clone()` Returns a shallow copy of this `ArrayList` instance.
`boolean`	`contains(Object o)` Returns `true` if this list contains the specified element. It uses the Object's equals(..) method to compare.
`E`	`get(int index)` Returns the element at the specified position in this list.
`int`	`indexOf(Object o)` Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. It uses the Object's equals(..) method to compare.
`boolean`	`isEmpty()` Returns `true` if this list contains no elements.
`int`	`lastIndexOf(Object o)` Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
`E`	`remove(int index)` Removes the element at the specified position in this list.
`boolean`	`remove(Object o)` Removes the first occurrence of the specified element from this list, if it is present. It uses the Object's equals(..) method to compare.
`int`	`size()` Returns the number of elements in this list.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

APPENDIX C : Sequence Diagram



END OF PAPER

a