CARDIFF UNIVERSITY
EXAMINATION PAPER

Academic Year:	2016/2017
Examination Period:	Spring
Examination Paper Number:	CMT205
Examination Paper Title:	Object Oriented Development with Java
Duration:	2 hours

Do not turn this page over until instructed to do so by the Senior Invigilator.

Structure of Examination Paper:

There are 5 pages.
There are 4 questions in total.
There are no appendices.
The maximum mark for the examination paper is **60** and the mark obtainable for a question or part of a question is shown in brackets alongside the question.

Students to be provided with:

The following items of stationery are to be provided:
ONE answer book.

Instructions to Students:

Answer THREE questions.
Students are permitted to introduce to the examination any textbook, any printed / handwritten notes, and other similar materials. Use of annotations, highlighting and bookmarks is permitted.
The use of calculators is permitted in this examination.
Important note: if you answer more than the number of questions instructed, then answers will be marked in the order they appear only until the above instruction is met. Extra answers will be ignored. Clearly cancel any answers not intended for marking. Write clearly on the front of the answer book the numbers of the answers to be marked.

The use of translation dictionaries between English or Welsh and a foreign language bearing an appropriate departmental stamp is permitted in this examination.

Q1. (a) What are the *values* of the variables `res1`, `res2`, `res3` and `res4` after the following sequence of Java statements has been executed? [4]

```
int inum = -4;
double dnum = 9.75;
double res1 = (int)dnum / inum;
double res2 = inum > 0 ? dnum : -dnum;
String res3 = inum + dnum + "abc";
StringBuffer buffer = new StringBuffer(res3);
buffer.replace(1, 4, "=");
String res4 = buffer.toString();
```

(b) Write Java statements for the following tasks:

i. Given a String `str`, create a new String with variable name `letterOnly` that keeps all the letters only and in the order they appear in `str`. [3]

ii. Generate a `double` array `numbers` and initialise it with 100 random floating point numbers in the range of 5 to 10. [3]

iii. Assuming `MyTask` is a class that implements the `Runnable` interface (but *not* a subclass of `Thread`), create an instance of this class using the default constructor and start running the code in a separate thread with the highest priority. [3]

(c) Assume you are developing a Java server program and you are writing a class `NetworkMonitor` that monitors all the network communication in your application. As only one instance of the class `NetworkMonitor` should be created, what design pattern will be useful in this scenario? Briefly give one approach to implement this design pattern (no more than three sentences). [3]

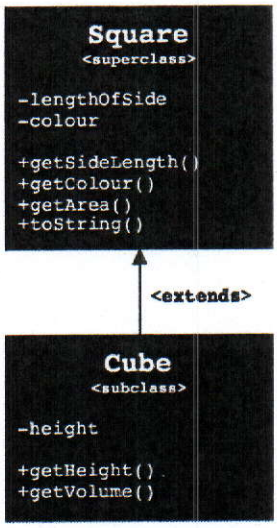
(d) Assume that a remote server implements the following interface

```
import java.rmi.*;
public interface CurrencyEx extends Remote
{
    public double exchangeRate(String from, String to)
                    throws RemoteException;
}
```

which returns the real-time exchange rate for the amount of the `to` currency equivalent to 1 unit of the `from` currency, where `from` and `to` are the currency code (USD for US dollars and GBP for British pounds). The RMI service is registered with a URL of
`rmi://rmi.cs.cf.ac.uk/currency`.
Complete the following code to print the amount of US dollars equivalent to 100 British pounds using RMI. [4]

```
import java.rmi.*;
public class CurrencyClient  {
   // main method
   public static void main( String[ ] args )  {
       // TODO: complete your code here
   }
}
```

Q2. (a) Consider the following class diagram:

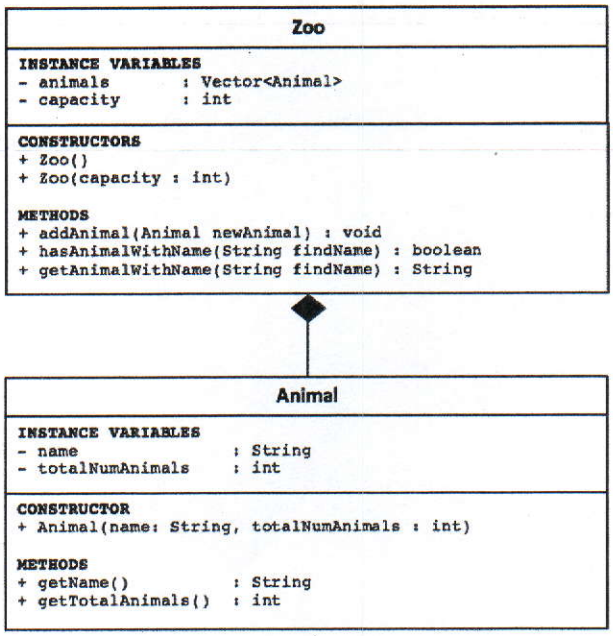


For each of the following concepts, explain how they work and using the classes Cube and Square (as defined in the class diagram above), provide a specific code example to demonstrate them:

- Substitutability
- Upcasting
- Downcasting

[12]

(b) The following class diagram represents the relationship between two classes: Zoo and Animal, through **composition** (i.e. the Zoo class is **composed** of the class Animal):



(Question continues on next page)

Given that composition exhibits a "has-a" relationship, we can say:

- A zoo is composed of one to many animals;
- OR, a zoo has one to many animals.

Detail how the Zoo and Animal classes could be **redesigned** to take advantage of object oriented features in Java, such as **inheritance** and **polymorphism**. The focus of your design should be to *promote code reuse* and to *separate the interface/implementation* to allow programmers that use your classes, to program at the implementation level.

NOTE: You are **NOT** required to provide an implementation, only a design. This can be in any form you deem appropriate (such as using a UML class diagram or a written commentary). [8]

Q3. Write a Java program `Sum` that reads a general text file containing integer numbers along with other general characters (e.g. letters). The user should provide only one command line argument, which specifies the text file name. Your program should find all the integer numbers in the text file, print these numbers (one number per line) and finally print the sum of all these numbers. An integer number is defined as a consecutive sequence of digits, separated by any non-digit character. You can assume that all the integer numbers in the text file and their sum can be represented using 32-bit integers.

Your program should include appropriate error handling. You can assume all the necessary classes from the Java standard library are imported.

The following skeleton program is provided and only the code to complete the program needs to be provided. [20]

```
import java.io.*;
public class Sum
{
    public static void main(String [] args)
    {
        if (args.length != 1)
        {
            System.err.println("One argument expected!");
            System.exit(1);
        }

        // TODO: Complete the program
    }
}
```

Q4. Complete the following Java program `LoginServer` which listens to incoming UDP packets at port 7000. Each packet contains a string containing a user name and a password, separated by whitespace. Every time a packet is received, the server program `LoginServer` sends back a packet containing a String containing either 'Successful' or 'Unsuccessful' depending on whether the username/password pair received matches one of the existing records. You can assume that the user account records are pre-loaded (code omitted for simplicity) into two String arrays (of the same length) `users` and `pwds` for usernames and passwords respectively. You should implement your server as efficiently as possible.

You may assume that the content received from the client is always in the correct format but communication exceptions need to be handled properly.

You can assume that relevant Java standard library classes have already been imported. The following code is provided and only the missing code needs to be completed:

```
import java.io.*;
import java.net.*;
import java.util.*;
public class LoginServer
{
    public static void main(String[] args)
    {
        // Code for loading users and pwds are omitted here.

        // TODO: Complete your code here
    }
}
```

[20]