OLLSCOIL NA hÉIREANN
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

Summer Exam 2010

Second Year Computer Science

CS2500: Software Development

Dr Carron Shankland
Professor J.A. Bowen
Dr M.R.C. van Dongen

INSTRUCTIONS: Answer **all 9** questions for full marks (225). All questions carry equal marks (25). There is **no need** to write whole classes *except where this is explicitly stated*. There is **no need** to write `import` statements. Use meaningful identifier names. Pay attention to the layout of your code and make sure your coding style adheres to the Java coding conventions. There is **no** need to add comments.

3 hours

Question 1: Basics. *(25/225 marks)*

(10 marks) 25

Question 1.a.
Provide a main( ) method that uses the enhanced for loop to print its arguments to standard output.

for (... : ...) [illegible]

(15 marks)

Question 1.b.
Provide a method that creates a Scanner to read from standard input. The method should use the Scanner to read all ints on standard input and print their sum to standard output.

Scan.hasnextint
scan.hasnext

Question 2: Classes. *(25/225 marks)*

(10 marks)

Question 2.a.
What is a class and how does it relate to an object?

blueprint instantiated

(15 marks) 20

Question 2.b.
Implement a class called NumberStream. The purpose of the class is to print ints to standard output. The class has an *instance method* void printNextInt( ) which prints the next number. The first number which is printed by printNextInt( ) should be 0. Subsequent calls to printNextInt( ) should print the sum of (1) the most recently printed number and (2) the current value of increment, which is a *class attribute*. The value of increment is initially 1. Your class should provide a setter method for increment. The setter method should be called setIncrement( ) and should be implemented as a *class method*. It is your task to provide the right type signature for setIncrement.

Make sure your class is implemented using good object oriented and software engineering standards.

Question 3: Inheritance. *(25/225 marks)*

(10 marks)

Question 3.a.
Explain the notion of inheritance, and state three advantages of inheritance.

(10 marks) 20

Question 3.b.
Provide a concrete example of how you may implement inheritance in Java. Explain your example in terms of the notions of 'being more specific' and 'being more general', and relate these notions to the notions of 'is-a', 'extension', 'subclass', and 'superclass'.

(5 marks)

Question 3.c.
Provide an example of overloading and an example of overriding. Explain the difference between overloading and overriding.

225

Question 4: Class Design. *(25/225 marks)*

Fota Wildlife Park has several `Animals`. Each `Animal` has *eating*, *roaming*, and *noise* behaviour. An `Animal` eats either grass or meat. Currently Fota has a `Hippo`, a `Lion` (`Feline Animal`), and a `Wolf` (`Canine Animal`). *However, they are expecting a `Dog`, a `Tiger`, a `Cat`, and many more `Feline` and `Canine Animals`.*

The `Hippo` eats grass and roams alone. The `Feline Animals` roam alone and eat meat. The `Canines` roam in packs and also eat meat.

Model the behaviour of Fota's `Animals` **using abstract classes in the higher levels and concrete classes at the final level of the class hierarchy.** The noise behaviour should be implemented by printing the name of the `Animal` that makes the noise. **Write each class and abstract class on a separate sheet in your answerbook.**

Keep in mind that your class design should support the addition of new `Animals` with as little coding effort as possible.

5-10

Question 5: Event Handlers. *(25/225 marks)*

Provide a class `ExamButton`. The class should have a `main( )` method, which displays a window with a button on in. The initial size of the window should be 200 × 300 pixels. Initially the button should display the text '`click me`'. However, when the user clicks on the button the text should change to '`Number of clicks =` n', where n is the number of times the button has been clicked.

Table 1 on Page 7 lists some methods and constants which you may use to answer this question. Notice that the table is not exhaustive and lists some red herrings.

0

Question 6: Recursion. *(25/225 marks)*

Question 6.a. *(10 marks)*

Explain the notion of recursion.

Question 6.b. *(15 marks)*

5-20

Implement a recursive method that multiplies two non-negative `ints`. The method is not allowed to use multiplication. *Hint: use repeated addition.* **Note:** No marks are awarded if you provide an iterative solution.

4 5

total = 5

```
public class PartialIterableClass /* FILL IN #1 */ {
    private String[] things;

    public PartialIterableClass( String[] things ) {
        this.things = things;
    }

    /* FILL IN #2 */
}
```

Figure 1: Contrived partial class.

Question 7: Iterators and Generics.

(25/225 marks)

Question 7.a.

(10 marks)

Figure 1 depicts a contrived partial class which implements the `Iterable` interface. The only purpose of instances of this contrived class is to provide a mechanism to iterate over the members of the attribute `things`.

This class can be completed by providing code for the '`/* FILL IN #1 */`' and the '`/* FILL IN #2 */`' in Figure 1. To answer this question you are asked to complete this class. Make sure you clearly indicate which code completes '`/* FILL IN #1 */`' and which code completes '`/* FILL IN #2 */`'.

Question 7.b.

(15 marks)

Provide a generic class `Pair` for representing pairs of things. The class should provide a constructor, a method for getting the first member of the pair, a method for getting the second member of the pair, a method for setting the first member of the pair, and a method for setting the second member of the pair. The class should be parameterised over two types: one for the first member and one for the second member of the pair.

Question 8: Exceptions.

(25/225 marks)

Joe uses his own `JoeConnection` class for making connections between his computer and other computers. The class provides the following constructor and instance methods.

`JoeConnection( String address )`: Make a connection with the URL `address`.

`void writeLn( String text )`: Write `text` to the `JoeConnection`.

`String readLn( )`: Read a line of `text` from the `JoeConnection`.

`void close( )`: Close the `JoeConnection`.

Joe's connections regularly fail and this causes errors.

Using proper exception handling, demonstrate how to use Joe's `JoeConnection` class to (1) make a `JoeConnection` with the URL `http://students.chat.box`. (2) write `"Hello world"` to the `JoeConnection`. (3) read in a string from the `JoeConnection`, and (4) close the connection. The exception handling should provide as many details as possible about the cause of failure, and print the stack trace which led to the failure.

Question 9: Enumerated Types.

(25/225 marks)

Question 9.a.

(10 marks) 10

Provide a critical comparison of `int` enums and Java enums. State advantages and disadvantages.

Question 9.b.

(15 marks)

Using Java enums, implement a class for payroll computation. The constants in the class correspond to the **normal** days of the week: Monday, Tuesday, ..., Friday, **weekend** days: Saturday and Sunday, and a Bank Holiday, which is a **special** day of the week.

The class should provide an instance method `double pay( double time, double payrate )` which returns the total pay of an employee who has worked on the current day. The rules for computation are as follows:

- The **pay** for the given day given by

$$\text{pay} = \text{base pay} + \text{overtime pay of that day} .$$

- Here **base pay** is given by

$$\text{base pay} = \text{pay rate} \times \text{hours worked} .$$

- The **overtime pay of that day** is given by

$$\text{overtime pay} = \text{pay rate} \times \text{overtime hours}/2 .$$

- The **overtime hours** depend on the kind of day.

Normal weekday: For a normal week day the overtime hours are the hours worked on that day in excess of 8 hours.

Weekend: For weekend days, the overtime hours are the hours worked on that day.

Bank holiday: For a bank holiday, the overtime hours are 1.5 times the hours worked on that day.

Make sure your class is maintainable: it should be possible to add and remove days without breaking existing code. *Hint: implement your class using the strategy enum pattern.*

Class/Interface	Methods and Constants
JButton	void addActionListener(ActionListener listener)
JButton	void addChangeListener(ChangeListener listener)
JButton	void doClick( )
JButton	void doClick(int pressTime)
JButton	void fireActionPerformed(ActionEvent event)
JButton	void fireItemStateChanged(ItemEvent event)
JButton	void fireStateChanged( )
JButton	void setText(String text)
JFrame	static int EXIT_ON_CLOSE
JFrame	Container getContentPane( )
JFrame	void setDefaultCloseOperation(int operation)
JFrame	void setSize(int x, int y)
JFrame	void setVisible(boolean visibility)
JFrame	void update(Graphics g)
Component	void add(Component comp)
Component	void addContainerListener(ContainerListener listener)
Component	void addPropertyChangeListener(PropertyChangeListener listener)
Component	void addPropertyChangeListener(String propertyName, PropertyChangeListener listener)
ActionListener	void actionPerformed(ActionEvent event)
ChangeListener	void stateChanged(ChangeEvent event)
ItemListener	void itemStateChanged(ItemEvent event)

Table 1: Useful methods and constants. Some of the `JButton` methods are inherited from `AbstractButton`. Some of these methods are not needed: they're red herrings.