



Complete, fold and seal

(see note 2 below)

SURNAME: NOVAK

FORENAME(S): Sophie

Examination Answer Books

12 Pages

Instructions to Candidates

1. Please give the information requested in the boxes below.
2. Enter your name in the space provided to the right, fold over and seal down (using the label provided) to hide your name.
3. The answers must be legibly written. The Examiners can either ignore illegible script or have it typed and the expense of typing charged to the candidate.
4. Write your answers in ink.
5. Begin each answer on a fresh page. Do all rough work in the answer book and cross it through.
6. No part of this book is to be torn out; it must be handed in complete.
7. If more books than one are used, fasten them together using the tags available from the invigilator(s).
8. **Every book must be handed in. In no circumstances may answer books, used or unused, be taken out of the examination venue.**

1	Student Number	P149298
2	Title of Examination Paper	Application Design and Development
3	Module Code	M5X01025
4	Date and Time of Examination	14/05/18 9 AM
5	Examination Venue	Mount Pleasant
6	Seat Number	
7	Number of Answer Books handed in	1

List the numbers of the questions you have attempted	Office Use
2 (a) (i)	
(a) (ii)	19
(a) (iii)	
(b)	
(c)	
(d)	
2 (a)	17
(b)	
(c)	
3 (a)	22
(b)	
(c)	
Office Use	77%

17?
✓
56
75
75%

1

(a)(i) Attributes contain the variables of a class they consist of the type (int, string) and the attribute name. An example of an attribute is: `private int - numberOne;` Encapsulation is the grouping of data together, it also has an access mechanism to limit how the values can be changed. The specific set of functions must be identified for changing it. Encapsulation is also the idea of bundling data and methods that work on data within one unit.

4

(a)(ii) Objects contain values and they are seen within the class. The relationship between a class and an object is that they are not similar, as classes do not contain values, only objects do but classes do define what information is held within the object.

4

(a)(iii) The features of inheritance is ~~the~~ parent classes are known as superclasses and child classes are known as subclasses. The main reason for inheritance is that the child gains the attributes and the methods from the parent class. Within inheritance is specialisation; as you move down the class hierarchy each class adds more specific attributes and or behaviours. There is also generalisation; as you move up the class hierarchy ~~each~~ the attributes and or behaviours become more general.

4

(b) Firstly there is a class named "Bark" which will output "Bark" onto the console each time it is called. There is another class called "House" which is where the guard dog attribute is assigned from the previous class the Dog. is then known also as dog which was through ~~the~~ using the constructor. The properties include when the BangDoor is called then the guard dog will ~~then~~ bark. Then finally in the methods the guard dog is also known

2.5

as dog.

~~2d~~ 1(d) at end.

- (c) The four mistakes shown in Appendix 2 are: within the Cat class, there should not be the word "void" in between "public" and "Cat" as this is a constructor not a property. As well as the word "string" within the brackets of the constructor ["public void Cat (string theCatName) there should not be a type within the constructor brackets, only with a property. The third and fourth mistakes are within the Owner class within the constructors section, in the brackets the word "Owner" should be the word "cat" as the Owner class is getting the information from the Cat class. Finally, the fourth mistake is not putting any argument declarations within the brackets of the method "GreetCat".

2



- 2 (a) The relationship between the "House" class and the "Owner" class is a one relationship and the "House" class and the "part" class is a ~~one to many~~ zero to many relationship. So the intended use of each class is the "House" only has one "owner" and the "owner" only has one "House". The "house" will have zero to many "parts" within.

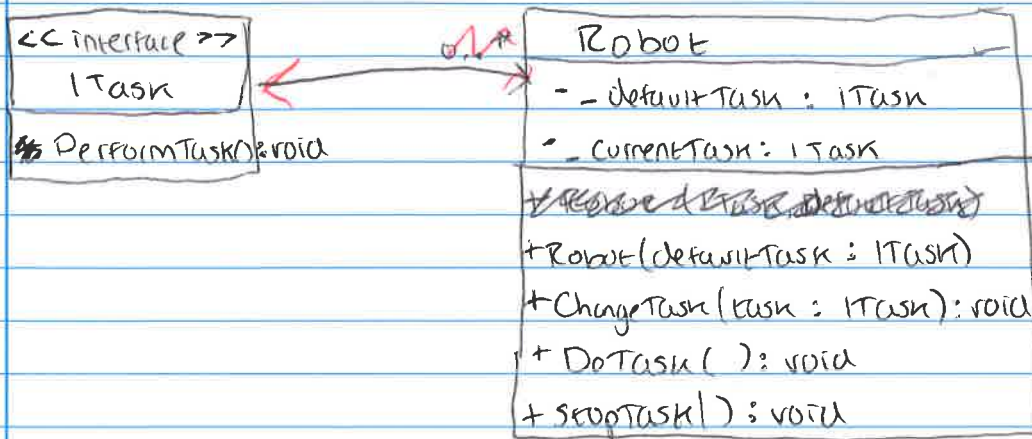
4



- 2(b) Abstraction has been used in Appendix 4 because it uses the parent class to provide everything for the child class. Abstraction is important because when the developer does not want to provide any code for a method in the parent class instead they want the child class to implement it then mark the method as abstract. The decorator pattern is being used within appendix 4, ~~because~~ I know this because it allows responsibilities to be added to an object dynamically instead of subclassing.

6

(c) The class diagram for Appendix 4 is shown below



3 (a) Single Responsibility Principle (SRP) means a class should fulfill only one responsibility, it should have one reason to change, it should do one thing and do that thing well. The main benefits are: It improves readability, reduces code complexity, reduces coupling and has high cohesion, which means it has one responsibility, so is specific.

(b) The main use of this application is for the user to make an order. They can make an order using either the mobile phone app or through the web. The user can then view their orders to see if they are all correct before placing the order and confirming it. Once the user sees the order is correct they can confirm the order which then must include the payment, within the interface. All the components have a specific and different relationship with the user. Also, the mobile phone app order and the web order have a relationship which is Make Order, as they can both be used.

(c) The black box testing goal is to be completely unconcerned about the internal behaviour and structure of the program.

Instead, it concentrates on finding circumstances in which the program does not behave according to its specification. Functional testing within black box testing tests ~~operations~~ ~~specific~~ specific behaviours or features and is given a set of inputs to check the outputs. Non-functional testing within black box testing tests operations ^{of systems} rather than a specific behaviour or feature. One ^{out of the many} systems black box testing uses is exhaustive input testing, this finds all the errors in the program and it makes use of every possible input condition as a test case. It reduces the number of test cases to smaller but just as effective set of tests, it also looks for ways to group similar inputs, similar outputs and similar operations together.

White box testing permits the developer to examine the internal structure of a program, this derives test data from an examination of the program's logic. Path testing within white box testing tests each path through the code, exhaustive path testing executes, via test cases, all possible paths of control flow through the program, then the program could possibly be completely tested. White box testing also can test using the debugger, within the debugger is data flow, which tracks a piece of data throughout seeing where it changes and how it changes. There is also error forcing which forces variables to specific values to cause errors to see if the code can handle them.

The reason to use black box testing is ^{because} ~~that~~ one strategy tests to fail, this purposely means being cunning and seeking to force bugs out and to deliberately break and fail code. It also uses boundary conditions, which for example: if a program only accepts one to ten characters, then the developer would test the boundaries of this. Such as testing firstly one and ten characters, but then testing zero characters, nine characters and eleven characters to see if the software has any bugs.

However, the reason to use white box testing is because it uses unit & / module testing, this is code which

Is designed to test on aspect of the source code, it tests sub-boundary conditions and tests formulae and equations. It typically consists of four phrases which are: setup, this declares required variables and creates required objects. Exercise, which is the code for performing the test.

Verify, which checks the results, usually using Assert() to flag errors. Finally, teardown which closes any resources opened for the test.

Overall if I was to choose one of these testing methods it would be black box testing, as it is more thorough.

10

1(d) Dog thedog = new Dog();

thedog.SetNoise = "Bark"

Console.WriteLine("The guard dog makes the noise 503", thedog.GetNoise)

1
2