

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.**

Complete, fold and seal
(see note 2 below)

SURNAME: MCCARTHY

FORENAME(S): BENJAMIN
NIALL

1	Student Number	
2	Title of Examination Paper	APPLICATION DESIGN AND DEVELOPMENT
3	Module Code	CSAC5001M
4	Date and Time of Examination	MONDAY 14 TH MAY 9:00
5	Examination Venue	PEMBROKESHIRE COLLEGE
6	Seat Number	3
7	Number of Answer Books handed in	

List the numbers of the questions you have attempted	Office Use
1	16
2	16
3	18
Office Use	50



1.

a)

data types

i) Attributes are features and values that a class has. For example, in a car class some of the attributes would be make, model and colour as these are characteristics that all cars have. Encapsulation is the process of combining data and functions into a single unit called class. With encapsulation, the data is not accessed directly; it is accessed through the functions within the class. These functions are known as getter and setter methods. These methods are public, whereas the attributes to be retrieved are private. ✓

(3)

ii)

An object is a representation of a real-world entity. These entities can be physical (car, dog etc.) or intangible (job, order etc.). The state of an object ~~can~~ be changed is represented by its attributes and can be changed or interrogated via the methods that can be performed upon the object. Objects are made up of attributes, constructors, methods and properties. A class is a factory for creating objects. The class is the logical expression of the code; and the ~~the~~ object is the physical existence of the data the code uses. ✓ multiple instances!

(3)

iii)

Inheritance enables objects to take on the properties (attributes and methods) of existing objects. A class ~~are~~ used as a parent class is known as a super / base class, whereas the child class

is known as a ~~the~~ subclass/derived class. Inheritance is used for reusability of code and can be used to create a hierarchy of classes that share certain sets of attributes. An example of inheritance could be a lion class which inherits from a mammal class, which then inherits from an animal class. Inheritance implies specialisation within the classes; as you move down the class hierarchy, each class adds more specific attributes and behaviours i.e. becomes more specialised. This also works in reverse for generalisation. ✓

objects from child classes can be used where parent class is used.

1 b) The class Dog is created and has a single method (Bark), which is public and returns the word "Bark" when called. The class House, instantiates a variable "guardDog" which is of type "Dog". A constructor is then called which takes the parameter "dog" of type "Dog", and calls the ChangeDog method. The ChangeDog method also takes the parameter "dog" of type "Dog" and sets the "guardDog" variable equal to "dog". The House class then has a method called "BangDoor()" which is public and returns no value. When this method is called, it will call the "Bark" method from the Dog class via the "guardDog" variable. parameter

(4)

1 c) In the constructor for Cat, the parameter should be "String name" as this is the value used within the constructor.

In the constructor for Owner, the type should be "Cat" rather than "Owner" as that is the type declared in the attributes above.

The GetName() method is never used within the Owner class to retrieve the name of a cat. Therefore the GreetCat() method would not work as a name has not been retrieved and cannot be used.

The GetName() method only returns the value "name" but doesn't actually ~~set~~ set it, i.e. it has the get functionality, but not the set functionality.

(3)

```
1 d) class Dog
    {
        //Attributes
        private string _name;
        private string _breed;
        private string _colour;

        //Constructor
        public Dog(string name, string breed, string colour)
        {
            _name = name;
            _breed = breed;
            _colour = colour;
        }

        //Methods
        public void Bark()
        {
            Console.WriteLine("Bark");
        }

        //Properties
        get
        {
            return _name;
        }
        set
        {
            _name = value;
        }

        get
        {
            return _breed;
        }
    }
}
```

```

set
{
    - breed = value;
}

get
{
    return colour;
}

set
{
    - colour = value;
}

```

```

class House
// Attributes

```

```

Dog - guardDog = new Dog("Max", "Collie", "Black");

```

```

private
// Constructor

```

```

public House (Dog dog)
{

```

```

    - guardDog = dog;
}

```

```

// Methods

```

```

public void BangDoor()
{

```

```

    - guardDog.Bark();
}

```

```

private void ChangeDog (Dog dog)
{

```

```

    - guardDog =
}

```

developed classes

rather than shown
how to use them.

2.

Directed

a) House has an association with Owner, the relationship being that every house has ~~only~~ one owner. House ~~also~~ also has an association with Part, where every house has zero to many parts. Car has a dependency relationship with Owner. A dependency relationship means one object depends ~~one~~ on another for its implementation. In this case, car relies on Owner. Window and Door both inherit from the Part class, so they will inherit the properties of the Part class. This would be used to create a hierarchical structure as the house would be made up of many parts, so by using inheritance and a hierarchy system it is likely that more code will be reusable and there would be less redundancy within the code.

(3)

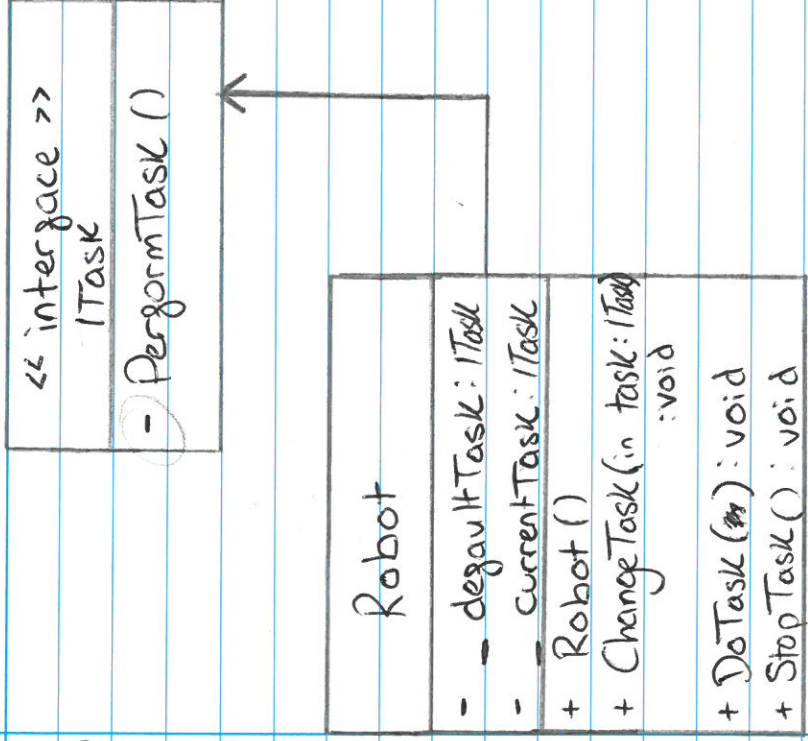
2.

b) The code in Appendix 4 uses the strategy design pattern. With this design pattern you identify and separate the aspects of the application that vary from those that stay the same. In this case, `PerformTask()` stays the same so this property is enforced ^{implemented} on everything that uses the `ITask` interface. ^{by other classes.} The main goal of abstraction is ~~to~~ to handle complexity by hiding unnecessary details from the user. This enables the user to implement more complex logic on top of the provided abstraction without worrying about the hidden complexities. Abstraction is useful as it improves the reusability of code and readability, as code that is redundant is not shown. This also removes the duplication of code and allows the user to focus on what the object does instead of how it does it. In the example, abstraction is created by the way of an interface. An interface is a structure that enforces certain properties on an object. This is useful because it allows the user to develop loosely coupled systems and in-turn makes unit testing and mocking easier. As well as this, it is also very useful for Dependency Injection.

(5)

m2

c)



⑧

3.

a) The Single Responsibility Principle states that a class should only fulfil one responsibility. It should one, and only one reason to change. Changes to classes should be heavily avoided as this could lead to errors. The Single Responsibility Principle improves readability as the name of a class and its methods should be more ~~var~~ informative because they describe a specific function that they perform. As well as this, it also reduces the code complexity due to every function being split up into different classes. This principle reduces coupling and improves cohesion, as each class only has one responsibility. Testability is also improved with this principle as it is a method carried out multiple tasks rather than one, it can be difficult to write a test for.

✓
(6)

3.b) From the diagram in appendix 5, it is evident that the program is intended to be used as an online store of sorts. The user is able to make an order through either a mobile phone app or via a web order form, both functions of which inherit from the main "Make Order" function. Users are also able to display orders. The user is able to confirm their order, however the "Confirm Order" function is dependant on the "Take Payment" function i.e. a payment must be taken before an order can be confirmed.

(4)

3.c) Black box testing is a testing technique where the person ~~using~~ testing the program has no knowledge of the inner workings of the application. It is intended to find circumstances in which the program does not behave according to its specifications. Black box testing entails functional and non-functional testing. Functional being specific behaviours and features are tested; and non-functional where just the operation of a system is tested rather than specific features. White box testing permits you to examine the internal structure of the program. This strategy derives test data from an examination of the program's logic. It is used to directly test low level functions or libraries and test software from the top level by adjusting test cases based on the code operation.

// Black box testing is useful because of ^{requirements} you able to test from the end-users point of view, which is important because they will be the ones to use it, not the creator. However, is the creator is performing these tests, it is easy for them to be biased by perhaps & subconsciously avoiding situations that may cause errors. Because of this, it would be more useful to use an external source for testing, however this is not always possible. Similarly, with white box testing it is easy to be biased and tailor the tests to match the code operation, rather than what it is required to do. It is

useful to use a combination of both Black and White box testing during the software development project. This is because, although Black box testing is useful, ~~the~~ certain parts are often over tested whilst other parts of the application are under tested. Whereas ~~white~~ White box testing is typically thorough, key end-user scenarios. By using a combination of both methods, it is more likely that all situations will be tested and reviewed. This could also be done via Gray box testing ~~where~~ which are design test with a customer focused perspective and uses a white box approach to ensure efficiency and test case coverage. ✓

⑧

Arholwr yn unig

Rhify Cwestiwn

Question No.

Examiner Only