|  |  |
| --- | --- |
| **Course** | Level 4 Diploma in Software Testing |
| **Course Code** | 603/5262/0 |
| **Unit Name** | Object Oriented Programming |
| **Unit Number** | CBE719 |
| **Assignment No.** | 1 |
| **Assignment Name** | The features and design of an OOP solution |
| **Assessor Name** | Kevin McLaughlin |
| **Internal Verifier Name** | Michelle Simpson |
| **Internal Verifier Signature** | Michelle Simpson |
| **Internal Verification Date** | 05/12/2022 |
| **Assignment Release** | 09/12/2022 |
| **Assignment Submission** | 13/01/2023 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Learning Outcomes (LO) Covered By Assignment** | | | |
| **LO 1** | | Understand object oriented programming and appropriate programming environments | |
| **Criteria to be Assessed (Tick if Achieved)** | | | |
| **1.1** | | **1.2** | |
|  | | | |
| **LO 2** | | Be able to design object oriented programming solutions. | |
| **Criteria to be Assessed (Tick if Achieved)** | | | |
| **2.1** | **2.2** | |  |

|  |  |
| --- | --- |
| **Learner Name** | Ryan McKee |
| **Date Submitted** | 15/01/2023 |

|  |  |
| --- | --- |
| **Assessor Comments** |  |

|  |  |
| --- | --- |
| **Assessor Declaration** | **I believe that the evidence submitted for this assignment is the learners own** |
| **Assessor Signature** |  |
| **Learner Declaration** | **I certify that the evidence submitted for this assignment is my own. I have clearly referenced any sources used in this work.** |
| **Learner Signature** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Assignment Verified (Y/N)** | |  | | |
| **Grade Agreed (Y/N)** |  | | **Signature** |  |

**Background**

Different Coatis a well-established paint and decorating business stocking all indoor/outdoor paints in particular. The business is open to the general public and trade customers.

Recently staff have been overwhelmed with a specific request from all customers. This is to determine how much paint is required to paint a specific room. Customers do not want to purchase too much or too little paint to complete a painting job. Customers want to know this initially before committing to purchasing specific volumes of paint. This information can be relayed to them on or after the calculation has been completed. Part of the solution requires you to design an object oriented solution to meet their requirements in relation to services being provided to customers.

**Complete the tasks that follow:**

**Task 1**

Initially they require an overview of object orientation to help understand the solution being developed and to aid future maintenance requirements. This should be informative and reflect the key areas of object orientation.

**Task 2**

Design the object oriented solution to include the following: **(See appendix 1 for guidance)**

* Class diagram created that illustrates all classes required with suitable detail
* Key screen design used for output of customer calculations
* Description of the objects to be created and data storage requirements. Include a relevant file/record structure with sample records to be used in the testing of the solution.

**Task 3**

Provide constructive reasoning behind the choice of Java as the development language being used to develop the design solution created. Make key references to the IDE being used.

**Note: Pay particular attention to assignment criteria provided when answering all tasks**

**Appendix 1**

**Note use appropriate object oriented techniques where appropriate. Please refer to UML Checklist discussed previously when designing class diagram.**

**Class Diagram**

**Base Class**

1. Design a base class that has the following information recorded about paint calculations:

* Customer Type
* Customer address
* Customer telephone number
* Surface area
* Room shape
* Height
* Width

1. Within the base class, design two constructors containing the following information:

* A constructor that will process customer type, customer address, telephone number and surface area
* A constructor that will process customer type, customer address, telephone number, room shape, height and width

1. Within the base class, design a method to be the base to work out the coverage for different customer types
2. Design necessary getter(s)/setter(s) for class
3. Within the base class, design a toString() method to aid testing and to allow further overriding within derived classes

**General Public Class**

1. Design a derived class that has the following additional information recorded about the general public

* Customer Name

1. Override the coverage method inherited from the base class.
2. Design necessary getter(s)/setter(s) for class
3. Override the toString() method from the base class

**Trade Customer Class**

1. Design a derived class that has the following additional information recorded about trade customers

* Trade Name

1. Override the coverage method inherited from the base class.
2. Design necessary getter(s)/setter(s) for class
3. Override the toString() method from the base class

**Constant Class**

1. Initially identify the key information required to work out the core calculation to paint different shapes of rooms. For clarity, room shapes will be square, rectangular and cylindrical. Note that all calculations will be in square feet. There will be constants used within the various calculations and these will be stored within a constants class.

**Helper Class**

1. Design a method within the helper class that will open and read the file created and store each record within an array list. Make sure this method has the capability to return this array list when required

**Output Screen**

Create an output screen that includes the following output:

* A detailed breakdown of calculation details including customer type, customer name, room shape, address, telephone number and paint calculation formatted
* A detailed breakdown of calculation details including customer type, trade name, address, telephone number and paint calculation formatted

**Assignment Criteria**

|  |  |
| --- | --- |
| **1.1** | Explain the principles, characteristics and features of object oriented programming |
| **1.2** | Critically evaluate the environment flexibility of programming in a given computer language |
| **2.1** | Select and justify choice of objects and data file structures required to implement a given solution design. |
| **2.2** | Create a design specification for an object oriented programming solution to a given problem |

**Task 1**

It’s important that Different coat understand the concept of Object-orientated programming as this will be the method of programming that will be used in order to design and build a solution that will help deal with the requests that have been overwhelmed staff being that customer want to know how many gallons/litters of paint that is need in order to paint there house so they are buying the correct amount of paint creating a program for this will create many benefits for the business for example added value to the customer who don’t want to be guessing how much paint they need to high customer service and potential resales of their products, On the other hand this will also quicken the process of finding out how much paint is needed for the staff freeing them up to do more valuable tasks for the business. Object orientation is going to allow for this solution to be created in a systematic and efficient manner for different coat.

Object-oriented programming (OOP) is a programming paradigm that is based on the concept of objects, which are instances of classes. OOP has several key principles, characteristics, and features, including:

Encapsulation which is the practice of hiding an object's internal state and behaviour from other objects. This allows objects to be treated as a black box, with their internal workings hidden from the outside world.

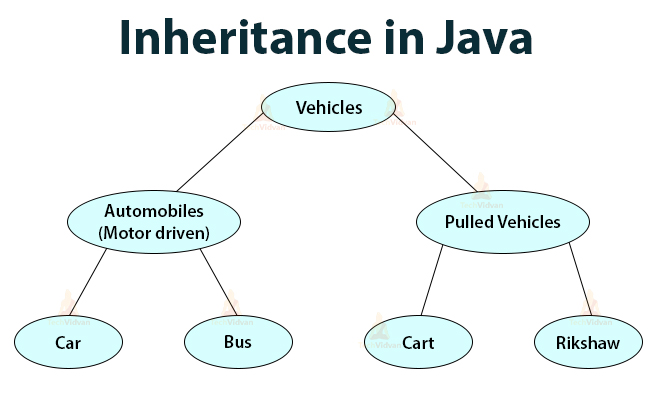
Text

Description automatically generated

A class is used to implement encapsulation by building fields and methods inside it so that outer classes cannot access and change fields and methods like in this example there is an outer class main which initializes an area object and class one of its functions to get the area however it cannot change or access anything else without the class being. This helps to keep related fields and method together. Which makes our code cleaner and easy to read and helps. Control the values of our data fields among other things.

Inheritance which is the ability for a class to inherit properties and behaviour from a parent class. This allows for code reuse and the creation of more specialized classes.

Below shows a diagram representing how vehicles inherit properties and functions and extend those functions so that specific vehicles can do their own specific tasks. For example, in this diagram car inherits from an automobile which inherits from vehicle therefore a car has the fields and functions of a vehicle and an automobile and a few more as well to make a car a more specific implementation of an automobile or vehicle.



Below is a code example of how inheritance works based on the diagram above this example shows the car class which inherits from automobile which inherits from vehicles. So, although in the car class I did not create the print Engine function and the move function when I call the print engine function move function on the car object they will print those methods defined in the super classes will be called.

class Vehicle {  
 String name;  
  
 void move(int n) {  
 System.*out*.println("moved " + n + " meters");  
 }  
}  
  
class Automobile extends Vehicle {  
 String engine;  
  
 public Automobile(String name, String engine) {  
 this.engine = engine;  
 this.name = name;  
 }  
  
 void printEngine() {  
 System.*out*.println(engine);  
 }  
}  
  
class Car extends Automobile {  
 int wheels;  
  
 public Car(String name, String engine, int wheels) {  
 super(name, engine);  
 this.wheels = wheels;  
 }  
  
 void printNumWheels() {  
 System.*out*.println(this.wheels);  
 }  
}  
  
public class Main{  
 public static void main(String[] args) {  
  
 Car car = new Car("Ferrari", "125cc", 4);  
 car.move(20);  
 car.printEngine();  
 car.printNumWheels();  
 }  
}

Text

Description automatically generatedThe third pillar of OOP is Polymorphism it gives the ability for objects of different classes to be treated as objects of a common base class. This allows for greater flexibility and code reuse.The code below shows an Animal base class and 2 other classes pig and dog which extend animal. In animal there is a function animal sound and pig, and dog extends that. Within the main there are 3 objects that are instantiated animal, bird and dog. When pig and dog are created the datatype of the object is animal this is because polymorphism within java allows this as pig and dog are still animals that both have their objects assigned to them even though the datatype is animal for them.

The fourth pillar of OOP is abstraction, This is the ability to focus on the essential features of an object and ignore the implementation details. This allows for the creation of simpler, more manageable code.

Abstraction example

in the main a pig object and two functions are called animalSound and sleep. For these functions to be called it is not essential to understand the underlying implementation of these functions you just need to know that they exist.

// Abstract class  
abstract class Animal {  
 // Abstract method (does not have a body)  
 public abstract void animalSound();  
 // Regular method  
 public void sleep() {  
 System.*out*.println("Zzz");  
 }  
}  
  
// Subclass (inherit from Animal)  
class Pig extends Animal {  
 public void animalSound() {  
 // The body of animalSound() is provided here  
 System.*out*.println("The pig says: wee wee");  
 }  
}  
  
class Main2 {  
 public static void main(String[] args) {  
 Pig myPig = new Pig(); // Create a Pig object  
 myPig.animalSound();  
 myPig.sleep();  
 }  
}

All of the pillars have been covered but there is other features of Java the first of these features are classes; these are blueprint’s for creating objects, which are instances of the class. Each object has its own state and behaviour and can interact with other objects through methods. This allows the implementation details of an object are hidden from other objects and can only be accessed through methods.

Classes and objects example

Below is an example of a plane class inside the plane class there is several fields and functions these fields include model, airline, engines and propPlane and the functions are a toString function which prints out the plane’s details. Within the test class the Plane object is created called plane it sets the values for each of the fields allows instance functions to be called on it for example the plane toString function.

class Plane {  
 private String model;  
 private String airline;  
 private int engines;  
 private boolean propPlane;  
  
 public Plane(String model, String airline, int engines, boolean propPlane) {  
 this.model = model;  
 this.airline = airline;  
 this.engines = engines;  
 this.propPlane = propPlane;  
 }  
  
 @Override public String toString() {  
 return "plane{" +  
 "model='" + model + '\'' +  
 ", airline='" + airline + '\'' +  
 ", engines=" + engines +  
 ", propPlane=" + propPlane +  
 '}';  
 }  
}  
  
public class test {  
 private static Plane *plane* = new Plane("Boeing 758", "Aer lingue", 2, false );  
  
 public static void main(String[] args) {  
 System.*out*.println(*plane*);  
 }  
}

Another important key feature of Java is Dynamic binding is when objects are only bound to a method call at run time, which allows for greater flexibility and ease of modification. One of the last features is Interfaces these are a way for classes to define a contract for methods that they must implement Method overloading and overriding Method overloading allows a class to have multiple methods with the same name but different parameters,

interface Animal {  
 public void animalNoise();  
 public void walk();  
 public void run();  
}  
  
class Cat implements Animal {  
  
 @Override public void animalNoise() {  
 System.*out*.println("meow");  
 }  
  
 @Override public void walk() {  
 System.*out*.println("Cat walks");  
 }  
  
 @Override public void run() {  
 System.*out*.println("Cat runs");  
 }  
}

The last features of OOP are Method overriding and overloading, Overriding allows a subclass to provide a different implementation of a method that is already provided by its superclass while overloading allows for the same function to exist multiple times however with different parameters.

Method Overloading example

class Main1 {  
 public void test(int num1) {  
 System.*out*.println(num1);  
 }  
  
 public void test(int num1, int num2) {  
 System.*out*.println(num1 + num2);  
 }  
}

Method Overriding example

class ReturnNumber {  
 public int returnNumber(int num) {  
 return num;  
 }  
}  
  
class ReturnDoubleNumber extends ReturnNumber{  
 @Override public int returnNumber(int num) {  
 return 2 \* super.returnNumber(num);  
 }  
}

Overall, the principles and characteristics of OOP, such as encapsulation, inheritance, polymorphism, and abstraction, provide a way to structure and organize code in a way that is easy to understand, maintain, and extend. OOP features such as classes, objects, and interfaces, allow for the creation of more complex and powerful applications like the one we are going to create for different coat.

Task 2

The first stage of developing this system is to provide the design for the project so that the requirements defined by Different coat can be fulfilled. There is a lot of design methods but for simplicity in this project we are going to use a UML class diagram this can be shown below.

Class diagram

Table

Description automatically generated

This is a UML class diagram each of the boxes shown above show an individual entity/class. Within these classes boxes there is information such as the class name, fields information and function information such as their names, data types whether they are static and/or final and the access modified whether they are public, private, package-private or protected.

My class diagram shown above has 7 classes The main class is the base this defines several attributes and functions pertaining to a customer including information fields for the type, address, telNo of a customer and information about the surface area roomShape height and width of a room. This class also has two constructors one for creating a calculating paint coverage using surface area and the other for calculating it with height and width of the room. The class also has 2 functions one for calculating the paint coverage of a room and the other for overriding the toString function so that the information about the class can be shown within a console. The display format for showing this information is shown below in the following screenshots:

**Key Screen designs**

**Main menu**

The main menu for this application will give the user two options they can either calculate the paint coverage for a new customer or they can read information about the paint coverage for an existing they will be able to input which option they want to do. Which will open a new console.

**Below is an example:**

Different coat paint calculation software

Please enter which option you would like to choose:

1. Calculate gallons of paint needed for new customer
2. Check details of existing customer.

Option selection: ||

If the user selects to calculate the gallons of paint needed for a new customer it will open some new options for the user as shown below, The user will be prompted to input the type of customer they are name address and telephone number and if the user wants to calculate the paint needed by height width and shape of the room or just by giving the surface area of a room.

Below shows an example of how the console would look like if the user choose to be general public and also choose the calculate how much paint would be needed for the specified surface area.

Please enter customer type:

1. General public
2. Trade customer

Input: 1

Please enter customer name:

Input: Kevin Hart

Please enter customer address

Input: 121 Darkley lane, Fermanagh

Please enter customer telephone number

07879424178

Please enter the surface meters surface area

Input: 123

Gallons of paint needed to cover 560 square meters: 1.4 gallons of paint

If the user was to select in the main menu to instead view the details of customers’ existing in the csv this is what would show up it would show information about each of the customer in the csv file.

Records in the database

customer name, address, telNo, surface area, roomShape, width, height, Gallons needed

Kevin hart, ‘121 Darkley lane, Femanagh’, 07879424178, 123, NA, NA, NA, 1.4 gallons

Two classes inherit from the base GeneralPublic and TradeCustomer, The difference between these two is that they add an additional field each GeneralPublic adds a string that will store the name of the customer, and TradeCustomer also adds a field which will store the name of the trade Customer.

In order to make each of the classes as understandable as possible I have separated as much functionality as possible into different classes this is what I have done for the Shape class. The shape class will be used to instantiate the roomShape field in the Base class this class contains an attribute for the shapeType which will be one of the values from the enumeration class . With the shape class there is also one more function which calculates the surface area for that specific shape. This will contain a switch statement where for an individual shape it will use the height and width passed in the this is not going to work I’m going to need to have parameters for this function that take the height and width if this is going to work.

As for the additional classes Shape will be a composite of the Base class and which will contain information about the shape and a function for calculating the surface area of that specific shape and the shape string will be one of the Enum values from the Shape Enum class which will contain all the shape options that different coat want worked on that were discussed in the requirements for this project. There is also going to be a Constants class this will contain constant values for calculations such as the value of PI and how much square foot a gallon can cover. Finally the helper class is going to be used for data storage it will allow for data about the calculations to be written to a csv (comma separated values) file, read from a csv file and allow for the information to be stored in an array List so that information is not lost when the program is closed. The reason I have chosen CSV as the storage system for this program is because it is human readable making it readable and easy to edit manually which makes it simple for the employees as different coat to manipulate and read records without the need for help. It’s also simple to implement and parse and has a small file size which will take up little resources of the devices that different coat. There are a few problems with csv however sincethis is a simple application they are not relevant and this is the best file system currently for different coats requirements.

**Below is an example of the way that data will be stored in csv files. This data will also be used for the testing of the applications read and writing functions:**

customer name, address, telNo, surface area, roomShape, width, height

Kevin hart, 121 Darkley lane, Femanagh, 07879424178,SQUARE,NA,20, 30

Bryan Cox, 19 Dublin road 07812345678,NA, 200, NA, NA

Dwayne Johnson, 40 supernova lane, 07812345681, CYLINDER, NA, 49, 124

Mike Williams, 109 Quark Road, 07312445381, 643,NA, NA,NA

Ham Spud, 46 Adelaide road high Wycombe, 0782345214, 432, NA, NA, NA

Task 3

During this project we will be using java to develop and design the software for different coat. Java is a popular choice for many projects due to its features such as platform independence, object-oriented programming, and strong support for enterprise applications. Java's platform independence means that code written on one platform, such as Windows, can be easily run-on other platforms, such as Linux or macOS, without modification. This will be useful for this project as it’s not statement by different coat what platform this software will be run on so it’s better this application have platform independence. Java's object-oriented programming features, such as encapsulation and inheritance, help to promote code reuse and maintainability. This is beneficial for Design coats as they may need to upgrade their solution in the future to fulfil further needs as the code will be simple to understand as there won’t be redundant code. Java also has strong support for enterprise applications, which are typically large and complex. Java also provides a large standard library and a wide variety of third-party libraries that can be used to perform common tasks, such as database access and web services or in different coats case we will need to use the Scanner package for getting user inputs and we will also need to use the File Reader and File Writer packages so the helper class can work as intended, In conjunction with Java IntelliJ will be used this is a popular Java development environment (IDE) that provides developers with a wide range of tools to help them write, test and debug code. Its features include code refactoring tools which automatically reorganizes and simplifies code which will help make code more readable and maintainable, code completion tools aiding developers to write code more quickly efficiently, and integrated debugging which allows developers to find and fix bugs quicker. Additionally, it also provides support for popular Java frameworks, such as Spring and Hibernate, which can be used to build enterprise applications quickly and efficiently.