QA Consulting Assessment Pack

**Exercise 1 - OOP Concepts**

* **Encapsulation**

Encapsulation wraps the data and code as a single unit. It keeps the classes hidden from other classes. Power steering is an example because the steering unit is independent and does not affect the functioning of any other mechanism. Encapsulation can be achieved by declaring the variables as private and by providing public setter and getter methods to view the variable values.

Below is an example of encapsulation:

public class Test {

private String name;

public String getName() {

return name;

}

public void setName(String newName) {

name = newName;

}

}

public class RunTest {

public static void main(String args[]) {

Test t = newTest;

t.setName(“Luke”);

System.out.print(“Name : “ +t.getName());

The setters and getters methods are the access points to the variable in the Test class.

* **Inheritance**

Inheritance is the process where a class acquires the properties of another class. Inheritance allows the information to be manageable in a hierarchical order. An example of inheritance would be a car class because a car inherits the properties of four wheels. The car also has sub classes that contain specific properties. Extends is the key word that is used to inherit the properties of an additional class.

Below is an example of inheritance:

public class Clothes {

public Clothes() {

System.out.println(“An item of clothing has been created”);

}

public class Shoes extends Clothes {

public Shoes() {

super()

System.out.println(“An item of shoes has been created”);

}

Using the extends key word allows the Clothes class to inherit methods from the Clothes class.

* **Polymorphism**

Polymorphism is the ability of an object to take on many forms. A use of polymorphism is when a parent class reference is used to refer a child class object. Gears on a car are an example because when the engine is accelerated then depending upon what gear is selected, movement is delivered to the car.

Below is an example of polymorphism:

public interface sport{}

public class football{}

public class player extends football implements sport{}

The player class is polymorphic because it has multiple inheritances.

* **Abstraction**

Abstraction is a process of hiding the implementation details from the user. The user will have the information on what the object does instead of how it does it. A car is an example because it is considered an object that contains many sub systems but the functionality is provided to the user.

Below is an example of an abstract class:

public abstract class Person {

private String name;

private int age;

public Person(String name, int age) {

System.out.println(“creating a profile”);

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}