Research Task

Association

Generic term to show relationship but not always ownership between two independent classes. The relationship can be different each case can be 1-to-1 1-to-many or many-to-many. The Association also can differ by being unidirectional or bidirectional meaning only one class knows the other or both classes are aware of each other and are able to interact.

e.g. Bank and Employees – Connected by objects

Two main forms of association:

Aggregation

Weaker relationship where the contained object can be independent to the main class.

E.g. Library and books as books can be removed from the library

Composition

A stronger relationship where the contained object is a **part-of** the container and the lifecycle is bound to the container.

E.g. Rooms in a house – If house is destroyed rooms are also destroyed

Method Overloading

Multiple methods in the same class with the same name. They have to differentiate by having different parameters. This can be done by different numbers of parameters or type of parameters. This will cause polymorphism to occur either compile-time/static.

Interfaces

Contract that will specify a set of abstract methods. It is like a set of rules. When class implements an interface, it needs to follow the rules.

e.g. remote control has different buttons similar to methods in an interface. – Different devices(classes) implement the interface. Each device decides how to perform the methods but buttons stay the same.

Glossary

Class – Blueprint that sets the properties and behaviours -

Object – An instance that has been created from a class with individual data and access to the class’s methods.

Data Fields – Variables inside of a class that sets the type of object it is e.g. int or string e.g. String name;

Methods – code that sets behaviour of an object – performs actions e.g. getName()

Constructor – Different type of methods to create objects, same name as the class and doesn’t have a return type Student s1 = newStudent();

Inheritance - A class will gain properties and behaviours from a different class e.g. class Prefeect extends Srudent {}

Overriding Methods – When a subclass extends a class but uses a new method with the same name that does a different job.

E.g. class Animal { void sound() { System.out.println("Generic sound"); } }

class Dog extends Animal { void sound() { System.out.println("Bark"); } }

Packages – A way to organise classes and interfaces that are going to intercat.

Modifiers - Change visibility or behaviour e.g. public,private,static,final

Polymorphism - A method/object taking many forms. 2 main types Compile-time and Runtime.

Encapsulation – Compiling data fields and methods into one class. Usually restrict access to data fields via private.

Abstraction – Only exposing essential features and hiding implementation details.