Principles of object orientated programming

Abstraction

Explain

Thinking about the class and designing it. What you need to create an object by breaking your problem down into parts.

Abstraction lets you focus on what the object does and how we define the structure of objects in the program.

Abstraction includes the thought process of taking something and breaking away unnecessary information.

Abstraction includes designing parent and child classes that can share common methods so that you can either generalize in the parent class so all the children classes can share common traits and so you don't repeat code.

Relate

We design classes by making UML diagrams to communicate the design with others.

Also another example of using abstraction includes making classes/methods abstract like we did in the spell book (task 12) we broke down the spells to their own child classes and we got them to override the methods from the parent spell class to perform specific commands.

Relationships

Abstraction is used to create plans for encapsulating a program by designing how the objects are structured. Abstraction is also used to determine where to use inheritance and Polymorphism in a program so that parent and child classes can share common methods.

Encapsulation

Explain

Encapsulation is what an object knows (fields) and what it can do (methods).

Encapsulation is enclosing an object into a class by placing the operations and data inside a capsule.

Encapsulation can involve hiding certain elements which protects information from outside the object by using private fields.

Relate

All the tasks we have done include encapsulation. Examples of this would be that the shape class knows it has a position on the x and y axis and the shape can use the method DrawShape to actually draw the shape.

Relationships

Encapsulation contains the principles for inheritance and Polymorphism within its class, as they generalize or specialize for what specific classes can know or do and what they can't know or can't do.

Inheritance

Explain

Objects have a lot in common so inheritance reduces duplication of code to generalise them to be shared. Child class objects get fields, properties and methods from parent class.

The specialisation child classes Inherit attributes and behaviour from a parent class.

Inheritance forms a relationship between the parents and children.

Relate

An example of this is the Shape Drawing task 11. Shape could be either a rectangle or a circle. Both of them would have a width, a position on x and y and a colour. It is important to leave the fields in the parent class to be used by both the circle or the square.

Relationships

Inheritance is used to design a program in a more efficient way by using object encapsulation to obtain what the parent class can know and do, for the program to perform tasks for the child classes.

Polymorphism

Explain

There are 3 kinds of polymorphism

1) Parametric: generics and templates

2) Ad hoc: function and operator overloading

3) Subtype: abstract class / interface

Most of our work will be using the Subtype polymorphism.

Polymorphism includes changing what a message/method does in a class.

You can create a method in the parent class without any actions and children classes can fill out the actions for their own unique specialisations.

Polymorphism brings flexibility, extensibility and adaptability to your object orientated programs.

Relate

In Shape Drawer Task 11 when we got the different shape child classes such as circle, rectangle and line we used override to change the DrawShape method from the Shape class.

Relationships

Polymorphism is used for Inheritance when designing parent and children classes to perform specific tasks for the child classes when the parent can't specialize for each unique child classes.

Polymorphism uses Encapsulation to access what the parent classes can do so it can specialize for the child classes.