Activity 3

Blake Wells

10/17/2021

Part 1

Text

Description automatically generated



A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

An Interface is an abstract type that acts as somewhat of a template for other classes. If a class extends an interface, those classes must include the same methods in the interface. Therefore you reach abstraction by extending an interface and while two classes may not be connected they still will have to maintain the same methods.

Part 2



Text

Description automatically generated



A screenshot of a computer

Description automatically generated

1. Diagram

   Description automatically generated

Polymorphism means the same code having different implementations based on the specific objects. In this activity the interface shape base was used to set a createArea function, each shape uses a slightly different createArea function. This allows for polymorphism in the code, each shape taking on its form YET, still allowing each shape to be processed through the same code.

Part 3



Text

Description automatically generated



Diagram

Description automatically generated

This code supports polymorphism by allowing one function to activate two separate classes through the use of an interface. This allows for different things to be done to specific objects while allowing them to be placed in the same array. This allows for more functionality than just a normal array of objects.

Part 4



Text

Description automatically generated



Graphical user interface, application

Description automatically generated



A screenshot of a computer

Description automatically generated

Graphical user interface, text

Description automatically generated

4.

Graphical user interface, text

Description automatically generated

Research Questions

1. What are some reasons a programmer might decide to declare a class as abstract? What are some reasons a programmer might decide to declare a class as an interface? Summarize your answers and rationalize them in 300 words.
   1. Abstraction allows for instantiation, an abstract class allows the programmer to hide details from the user, It also allows for one set of code to apply to many other classes that don’t directly drive from the same class. One example is the bomb and gun, both have the same method derived from the interface and thus both can be compiled into the same array of objects and used, this allows for the polymorphism of a singular class method.
   2. An abstract class is a class that cannot be used to create objects, it can only be inherited from another class. This allows for multiple classes to inherit the abstract class, use its methods, and remain separate classes with their unique methods. An interface also allows for abstraction, except instead of inherited the class is implemented, it has no workable functions but acts more like a template for other classes, allowing the compiler to know these are to be taken as the same function even though they are separate classes.
   3. A programmer might decide to use abstract classes over interfaces if they want a direct inherited link between the classes, where a class will use an abstracted method while the abstract class cannot be created itself, this is seen in the superhero code, where both heroes use the same abstract class methods, so it makes sense to allow for them to be inherited from the base class. In inheritance the goal is to set a template, that is then reworked into other classes, this interface may store the variables, but it does not store the methods, only defines them together as the same method. This is useful in the shapes code as each shape has a different calculate area, but they all want to be processed as if they have the same method. Thus, an interface is used.
2. What does the keyword final do on a class member variable? What does the keyword final do on a class method? What does the keyword final do in a class? Why would you want to mark a class or method static? Summarize your answers and rationale in 400 words.
   1. The final keyword is used to restrict whatever it is placed on, when a final is used it cannot be modified or overridden, it will operate exactly as it is implemented in the class. It can be inherited only the value cannot be modified. This is useful in creating various constants, and prevents inheritance and overwrites. A final class cannot be extended, it acts as the final stop in the class.
   2. The final keyword on a method cannot be overridden, a final method cannot be changed by a child class. This allows a programmer to maintain control over how their code is used. Say one has a specific method for a test machine and this method must operate in a specified fashion or wrong values may occur. This is a good use case for the final method. Anything that uses this method cannot overwrite it, meaning nobody can come along and reimplement the method and mess it up.
   3. A class with the keyword final cannot be inherited, the class acts as a final implementation of the class. This blocks other classes from unwittingly using its methods and allows for greater control over which methods utilize this class. A programmer now knows if the class is final it cannot be overridden and is called directly from another class.
   4. If a method is labeled as static it means it directly belongs to the class instead of an instance of the class. This means when an object is made from the class the method does not belong to the object but belongs to the class in which the object originates. This also means that an object's method can be utilized without calling a new object of the class. This allows for greater control over classes by allowing programmers to decide when their code is called and by whom.
   5. A static class is similar to an abstract class, it is essentially a class of methods but not objects. A static class cannot be inherited or instantiated and all members of the class are by default static. This allows a programmer to organize different methods into classes without needing to create individual objects. The methods can be called directly by anybody. This allows for the programmer to not need to implement convoluted object switching and temporary variables to utilize class organization of methods. This creates a cleaner environment that can use methods from other classes without having direct objects of the class.