Assignment #6: Polymorphism

Rowan Pilon

041049454

For CST8284

Due October 24, 2023

**Task #1:** Write a java code to illustrate the concept of run-time polymorphism and generate the appropriate output.

I accomplished this by creating 3 classes, named: runtime, runtimechild, and runtimechildtwo. I gave runtime a method called message() and I made runtimechild and runtimechildtwo children of the runtime class. In their classes, runtimechild and runtimechildtwo made re-iterations of the message() method, simply changing the output. In my main method, I create an object of each class and call its message() method, which results in a different output each time, as a result of runtime polymorphism and method overriding.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Task #2:** write a java code to illustrate the concepts of compile-time polymorphism

I accomplished this task via method overloading, I created a class compile and gave it a method average(int a, int b), I then created another method of the same name, but with different parameters, average(int a, int b, int c). This results in compile-time polymorphism via method overloading.

A screenshot of a computer program

Description automatically generated

**Task #3:** write a program to use abstact method and abstract class in Java.

To accomplish this task, I created three classes, the first one named taskthree. Taskthree is an abstract class with an abstract method called flavor, since this method is abstract it has no body. My second class is lemon, which is a public class that is a child of the taskthree abstract class, and therefore has access to the flavor method. So I created a body for the flavor class inside of lemon, which printed how lemons taste. My third class is sugar, which again is a standard class and child of taskthree, sugar also adds a body to the flavor function, which prints how sugar tastes. Finally, I used the main method to create objects of lemon and sugar respectively, and then had them call upon their iterations of the abstract method, flavor(). I believe this demonstrates the nature of abstract classes and their inability to be instantiated, as well as an abstract methods inability to have a body during their initial definition, but are able to have one thanks to inheritance to non-abstract child classes.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated