Name: Ali Albayrak

Student ID: P304320

Practice Activities 2.2

Vocabulary:

Downcasting: The type of casting that changes a generalized object to a more specialized object type.

Overriding: The process of a call to a generalized method and actually calls the instantiated subclass method, or appropriate subclass method.

Instanceof: The operator that allows you to compare a class instance against a class type.

explicit typecasting: The process of explicitly changing one data type to another data type.

Abstract class: A class with an abstract constructor and at least one method that is defined but not implemented.

Upcasting: This type of casting changes a specialized object instance into a generalized instance. It doesn't lose any of its detail but you can't access them without downcasting the object to access specialized methods.

Abstract class constructor: A constructor without implementation that makes the class restricted in that it cannot create instances.

1.Give one reason why you might use an Abstract class rather than an Interface

If we expect that classes that extend our abstract class have many common methods or fields or require access modifiers other than public (such as protected and private), we might use Abstract class rather than an interface

3. What would the output of the following be? Explain your answer.

In the sample code, we are creating two classes; animal and dog. Dog class extends animal class. So it has the access the animal class methods.

By this code;

Animal animal = new Animal();

We are creating a Animal named animal from Animal() class.

In the next code;

animal.makeNoise();

We are calling the makeNoise() method inside Animal class. Because we created animal from there. So output should be “talk”.

In the next code;

Dog dog = new Dog();

We are creating a Dog named dog from Dog() class.

When we run this code;

dog.makeNoise();

We are calling the makeNoise() method from Dog class. Because our object is from dog class. So output of this should be “Bark”.

In the next code;

Animal animaldog = new Dog();

We are creating a Animal named animaldog from Dog() class. So output of this should be “Bark”.

4.Using the animal and dog classes above. If we added the following code to the driver what would the output be:

**if** (animal **instanceof** Animal)

System.***out***.println("animal is Animal");

**if** (dog **instanceof** Animal)

System.***out***.println("dog is Animal");

**if** (animaldog **instanceof** Animal)

System.***out***.println("animaldog is Animal");

**if** (animal **instanceof** Dog)

System.***out***.println("animal is Dog");

First three of if sentences should return true. So output must be

animal is Animal

dog is Animal

animaldog is Animal

5. Describe casting.

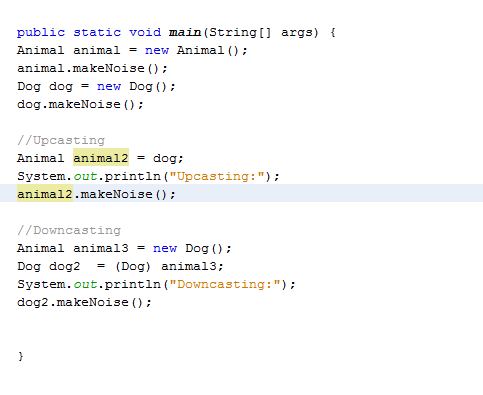
Casting is the principal of changing the object type during assignment. There are two types of casting; Upcasting and Downcasting.

Upcasting loses access to specialized methods in the subclassed object instance.

Downcasting gains access to specialized methods of the subclass.

6. Using the animal and dog classes above. Show examples of using a downcast and an upcast.

Upcasting:



This is good example for upcasting. We are creating an animal2 from a subclass dog. So animal2 loses its access to animal class because it is from dog class.

Downcasting:

As we can see from the screenshot above, we created an animal named animal3 from Dog() class in the downcasting section. If dog class has an another method inside, with downcasting dog2 can use it while, animal2 cant.

|  |
| --- |
|  |