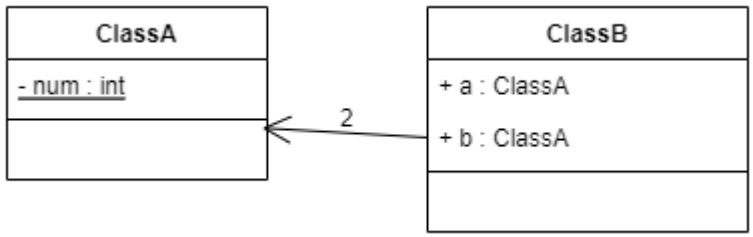
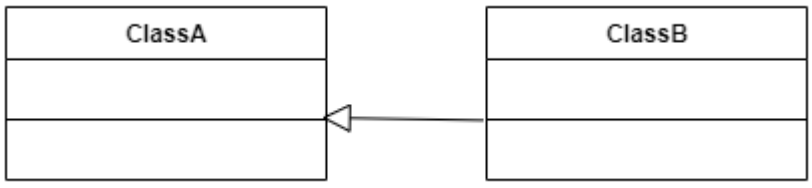
Original Partner **SOLUTION** Student #	Obstacle 6 COMP 10062	Paired Partner ("grader") Student # **Inheritance**	SCORE OBTAINED 5

1	**public class ClassA {** ** private static int num;** **}** **public class ClassB {** ** ClassA a, b;** **}**	1a) What kind of a relationship are ClassA and ClassB in? **ASSOCIATION i.e. a "has a" relationship** _____ type of relationship 1b) Draw the UMLs (with the correct arrow) which describes the two classes. 
2	**public class ClassA { }** **public class ClassB extends ClassA { }** 2a) What kind of a relationship are ClassA and ClassB in? **Inheritance, i.e. a "is a" relationship** _____ type of relationship 2b) Draw just the connecting arrow between these two UMLs.  	

3 Provide a small example of OVERRIDING and one of OVERLOADING methods

//OVERRIDING EXAMPLE

```
class Animal {
   void makeSound() {
      System.out.println("Animal makes a sound");
   }
}

class Dog extends Animal {
   @Override
   void makeSound() {
      System.out.println("Dog barks");
   }
}
```

In Java inheritance, overriding refers to the capability of a subclass to provide a specific implementation of a method that is already defined in its superclass. When a subclass provides its own implementation of a method that is already present in its superclass, it is said to be overriding that method.

//OVERLOADING EXAMPLE….SHOWING 4 OVERLOADS for makeSound

```
class Animal {
    void makeSound() {
        System.out.println("Animal makes a
sound");
    }

    void makeSound(String sound) {
        System.out.println("Animal makes a " +
sound);
    }
}

class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Dog barks");
    }

    void makeSound(int numBarks) {
        for (int i = 0; i < numBarks; i++) {
            System.out.println("Dog barks");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Animal animal = new Animal();
        animal.makeSound(); // Output: Animal
makes a sound
        animal.makeSound("loud noise"); //
Output: Animal makes a loud noise

        Dog dog = new Dog();
        dog.makeSound(); // Output: Dog barks
        dog.makeSound(3); // Output: Dog barks
Dog barks Dog barks
    }
}
```

In Java, method overloading refers to the capability of defining multiple methods in the same class or in a parent class with the same name but different parameter lists. That is, overloaded methods have the different method signatures.

1. ***Same Method Name:** Overloaded methods have the same name within the class.*
2. ***Different Parameter Lists:** Overloaded methods must have different parameter lists, which can vary in terms of the number of parameters, their types, or their order.*
3. ***Return Type:** Overloaded methods can have the same or different return types.*

)

4

Declare a 1000 element array of Dice. Then, roll each die once.
Output only the current face on the 5th die of the array.

Die
- numberOfFaces: int - currentFace: int
+ Die( ) + Die(numFaces: int) + getCurrentFace( ) : int + roll( ) : int + toString( ) : String

```
Die[ ]  d = new Die[1000];
for(int i=0; i< d.length; i++) {
    d[i] = new Die( );
    d[i].roll( );
}

System.out.println(  d[4].getCurrentFace( ) );
```