OOP Exam #1 3/2/2021

#1

@

1) f version AA output

2) k version AA output

3) f version AA output

4) BB method AA output

5) BB method AA output

6) BB method AA out put

7) 300

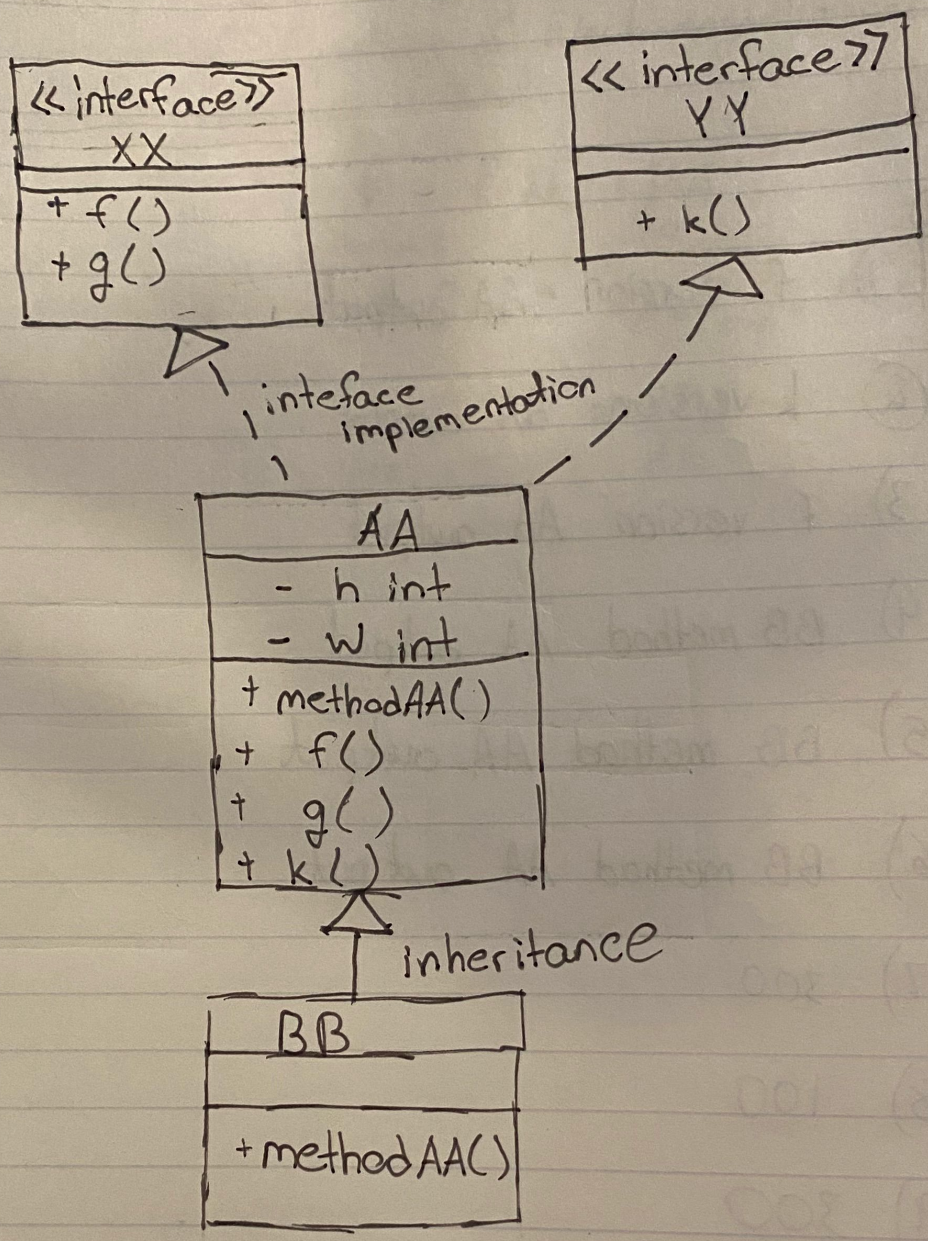
8) 100

9) 300

10) The program will run and execute normally.

The original constructor was written that way because it is a clearer way of showing what happens with the parameters.

b)


«interface»
XX
+ f()
+ g()
«interface»
YY
+ k()
inteface implementation
AA
- h int
- w int
+ methodAA()
+ f()
+ g()
+ k()
inheritance
BB
+ methodAA()

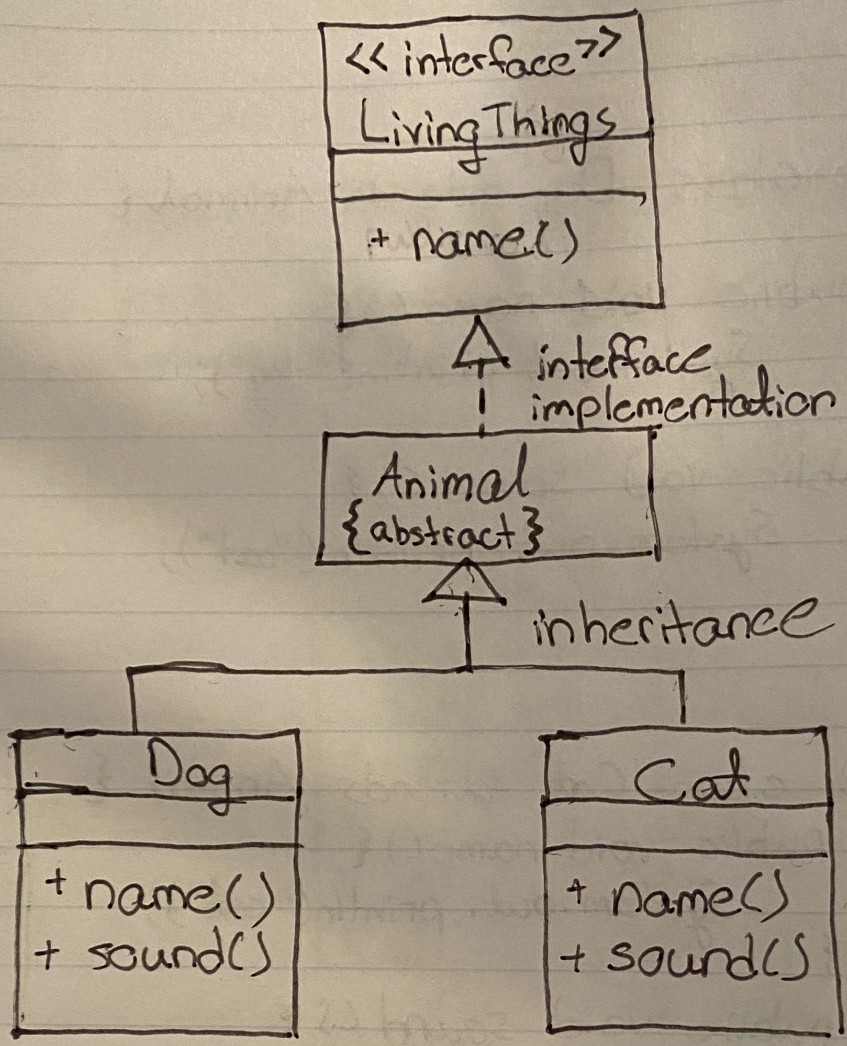
(2) @

```
public class Dog extends Animal {

    public void name(){
        System.out.println("dog");
    }
    public void sound (){
        System.out.println("cat");

    }
}
  public class Cat extends Animal {
      public void name(){
          System.out.println("cat");
      }
      public void sound (){
          System.out.println("meow");
      }
}
```

(b)


<<interface>>
LivingThings
+ name()
interface implementation
Animal
{abstract}
inheritance
Dog
+ name()
+ sound()
Cat
+ name()
+ sound()

#3

(a)

```
public int compareTo(ComplexNumbers w){

    return (int)(Math.signum(((re*re)+(im*im)) -
      ((w.getRe()*w.getRe())+(w.getIm()*w.getIm()))));

}
```

This is a very compact format that I split into 2 lines.

(b)

2.0 - 3.0I
-3.0
2.0 + 4.0I
3.0 + 4.0I
14.0I
18.0 -1.0I

(c)

```
public class ComplexNumbersByProduct implements
                                    Comparator<ComplexN

{

    public int compare(ComplexNumbers z, ComplexNumbers w
        return (int)(Math.signum((z.getIm() * z.getRe())
            - (w.getIm() * w.getRe()));

    }

}
```

(d)

18.0 - 1.0I
2.0 - 3.0I
-3.0
14.0I
2.0 + 4.0I
3.0 + 4.0I

(e)

```
public int compare(ComplexNumbers z, ComplexNumbers w)
{
    return (int) ((z.getIm() == w.getIm()) ?
        Math.signum(z.getRe() + w.getRe()):
        Math.signum(z.getIm() - w.getIm()));
}
```

↖ Used a ternary expression to condense the code

(f)

2.0 -3.0I
18.0 -1.0I
-3.0
2.0 +4.0I
3.0 +4.0I
14.0I