#### Questions

Once this exercise has been finished, answer the following questions:

* Which methods can be really invoked on the collection elements?

Depending on whether the element is a queen or a castle, it can be invoked by its corresponding methods in addition to the methods inherited from the Figure class.

* In the queen and castle classes you can invoke the WhoAmI movement methods to know the name of the object.If Castle class has implemented void castle() method, could it be possible to invoke that method from a reference to the base class? Why?

It is not possible because the base class does not inherit from the derived classes and does not know that method.

* What should we have to do in order to be able to use the previous void castle() method from an object of Castle class that is pointed by a reference to Figure class?

We should do a casting to the object that is being referenced by the Figure class and cast it to Castle as long as it is an instance of Castle.

* What should we do to know exactly to which class belongs every object pointed by a reference to the base class?

Ask if the object is an instance of some derived class using the “instanceof” operator

**RECTANGLE**

* + Show the difference between a class and an object.

A class is an abstraction of a real-world object, it is like a mold or template, and an object is an instance of a class.

* + Which steps are involved in the instantiation's process of an object?

To instantiate an object, we just have to have created the class, and then we create an object of that class by calling its constructor.

* + How is an object instantiated in Java?

Is an object that can have particular attributes and can have a particular behavior.

**SQUARE**

* + What is inheritance?

The inheritance is a mechanism by which a base class can inherit its attributes and methods to other classes called derived classes.

* + How do you express in Java that one class *inherits* from another?

With the keyword “extends”, the base class must have this keyword in its declaration of class.

* + Which methods of the superclass are visible from the subclasses?

The public, protected and default methods, in this case the getFigureType, setFigureType, area y drawTxt, methods of class Rectangle are visible to its subclases.

* + What is the meaning of method overriding?

The @Override annotation is used to indicate that a method from the base class is being overwritten in the derived class.

**REFERENCES TO INTERFACES**

* What type are both instantiated objects (in options 1 and 2)?

For option 1 it is a rectangle and for option 2 it is a square that inherits from the rectangular class

* What type is the variable that refers to them?

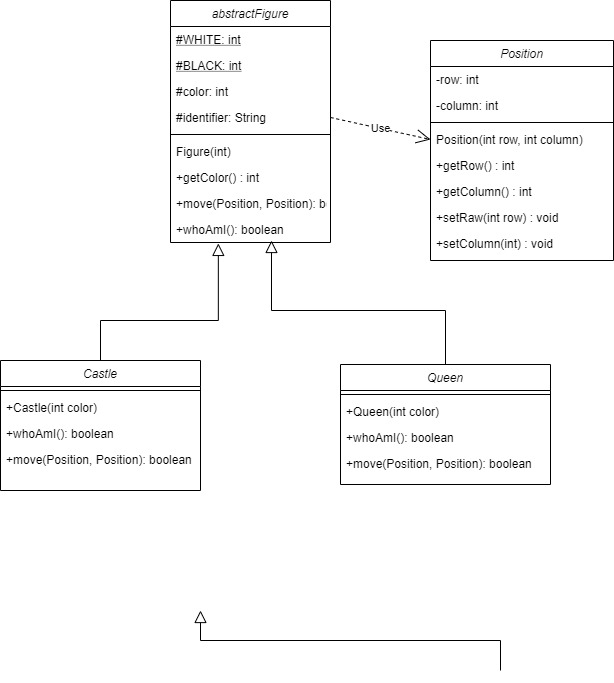
Is of type GeometricFigure, but depending on the option that the user selects we instances a rectangle or a square.

* Which methods of the superclass are visible from the subclass?

All public, protected and default methods, in this case the setDescription, getDescription, setType, getType, move, printDescription, getTag y getArea methods of class GeometricFigure are visible to its subclases.

* Can the same variable be used as a reference for different types of figures? Why?

Within the same context you can't use the same name even if they are of different type, but in different contexts you can use them, you have to take into account in variable's scope, each variable name must be unique within its scope

DIAGRAMS CLASS

