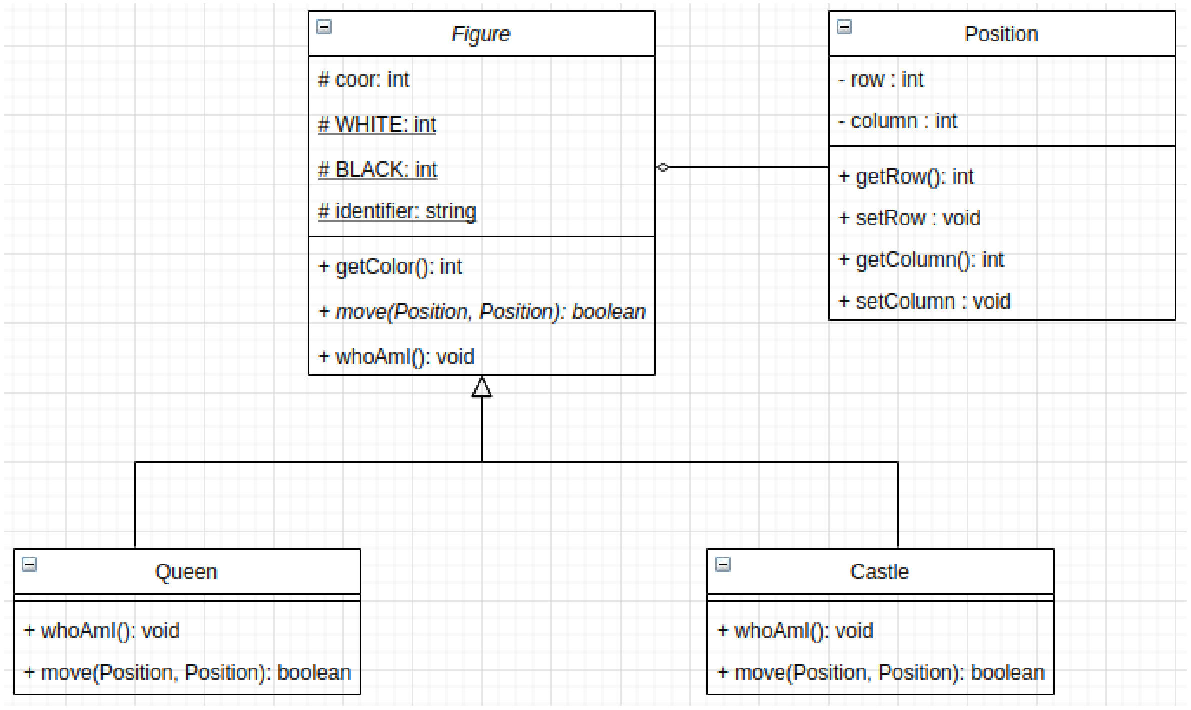
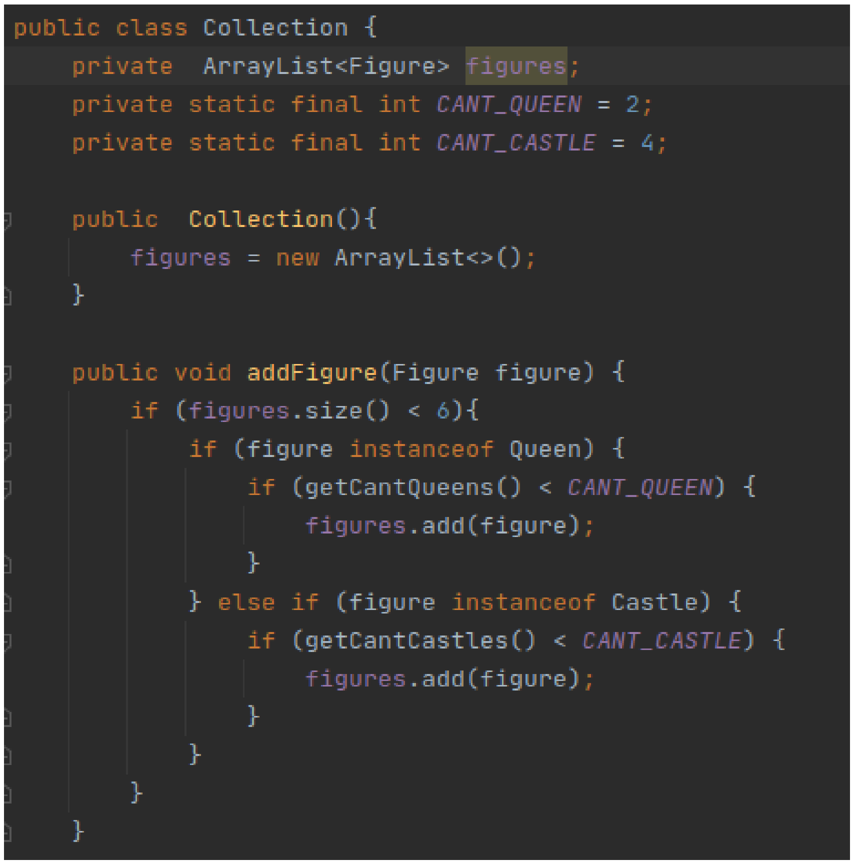
Exercise 1 Inheritance and references to the base class

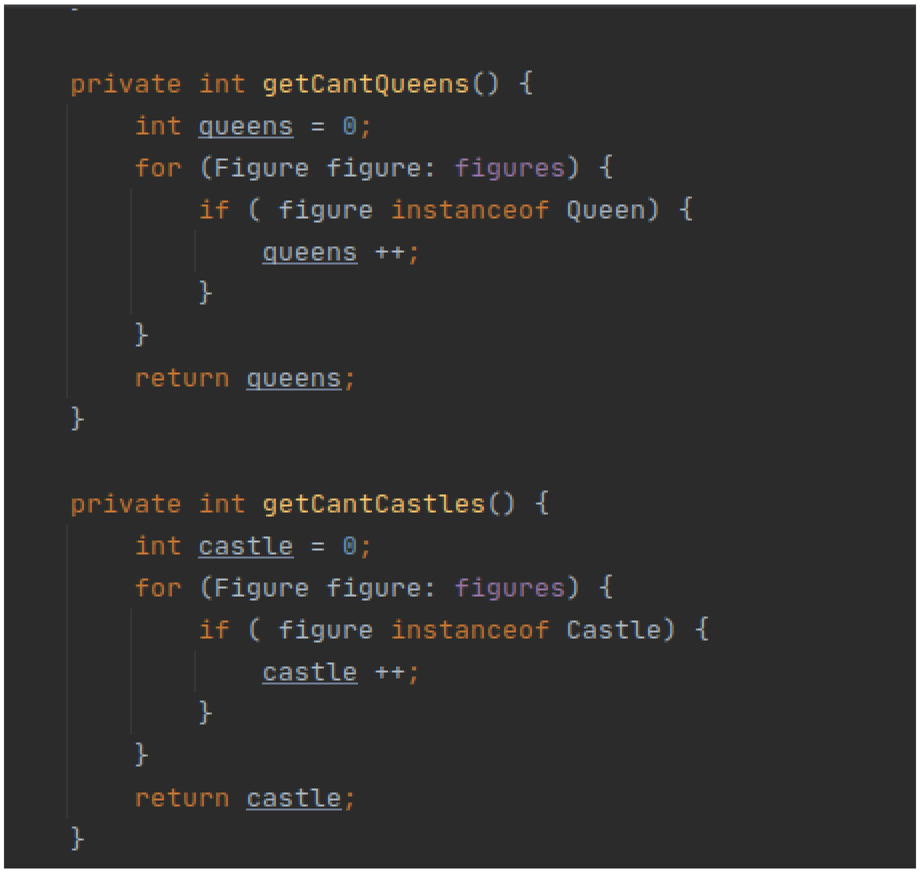
Class Diagram initial



Implement the necessary code to hold, in the same collection and indistinctly, 2 objects of Queen class and 4 objects of Castle class.

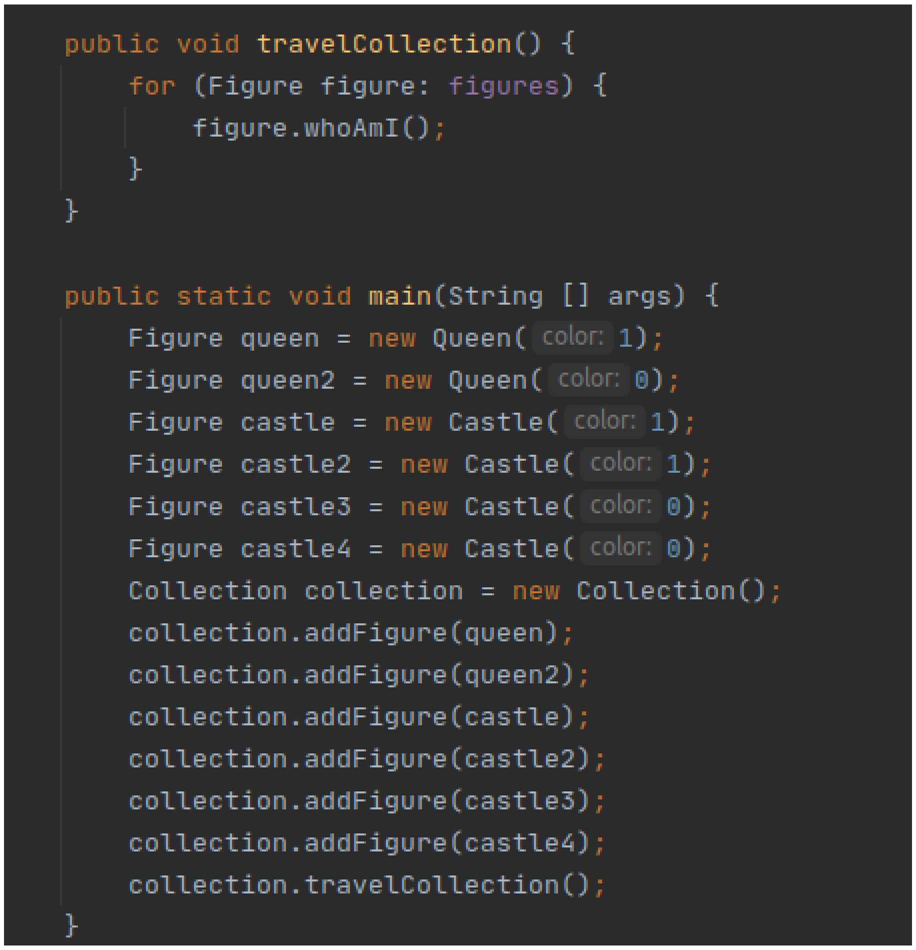


```java
public class Collection {
    private  ArrayList<Figure> figures;
    private static final int CANT_QUEEN = 2;
    private static final int CANT_CASTLE = 4;

    public  Collection(){
        figures = new ArrayList<>();
    }

    public void addFigure(Figure figure) {
        if (figures.size() < 6){
            if (figure instanceof Queen) {
                if (getCantQueens() < CANT_QUEEN) {
                    figures.add(figure);
                }
            } else if (figure instanceof Castle) {
                if (getCantCastles() < CANT_CASTLE) {
                    figures.add(figure);
                }
            }
        }
    }
```



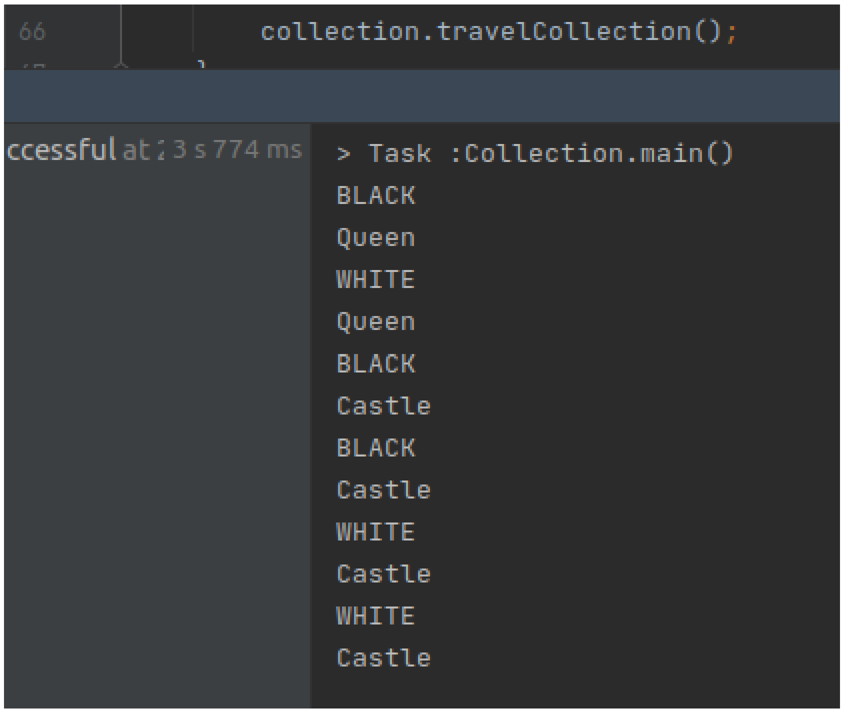
```
private int getCantQueens() {
    int queens = 0;
    for (Figure figure: figures) {
        if ( figure instanceof Queen) {
            queens ++;
        }
    }
    return queens;
}

private int getCantCastles() {
    int castle = 0;
    for (Figure figure: figures) {
        if ( figure instanceof Castle) {
            castle ++;
        }
    }
    return castle;
}
```

Once the previous code has been implemented, traverse the collection invoking public void whoAmI() method to test the correct operation of the base class references.



```
public void travelCollection() {
    for (Figure figure: figures) {
        figure.whoAmI();
    }
}

public static void main(String [] args) {
    Figure queen = new Queen( color: 1);
    Figure queen2 = new Queen( color: 0);
    Figure castle = new Castle( color: 1);
    Figure castle2 = new Castle( color: 1);
    Figure castle3 = new Castle( color: 0);
    Figure castle4 = new Castle( color: 0);
    Collection collection = new Collection();
    collection.addFigure(queen);
    collection.addFigure(queen2);
    collection.addFigure(castle);
    collection.addFigure(castle2);
    collection.addFigure(castle3);
    collection.addFigure(castle4);
    collection.travelCollection();
}
```

Output



```
66          collection.travelCollection();
```

```
ccessful at ? 3 s 774 ms   > Task :Collection.main()
                           BLACK
                           Queen
                           WHITE
                           Queen
                           BLACK
                           Castle
                           BLACK
                           Castle
                           WHITE
                           Castle
                           WHITE
                           Castle
```

Further questions

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

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

All methods of the Figure class or super class can be invoked.

● **If Castle class has implemented void castle () method, could it be possible to invoke that method from a reference to the base class? Why?**

Form the Figure class it's not possible to invoke the void castle () method, because this method is property of the Castle class, Figure class only can access its methods. In inheritance the super classes can't access the attributes and methods of the subclasses, while subclasses can inherit all the methods and attributes of super class.

● **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?**

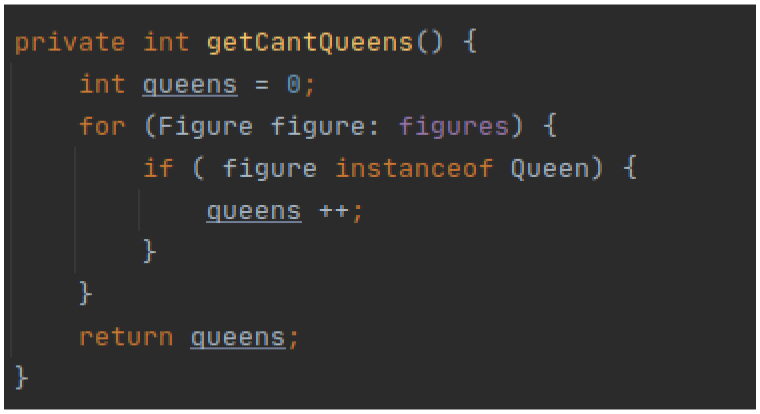
Using polymorphism.

1. We can create void castle () method in the Figure class, and overwrite this method in the Castle subclass.
2. In Figure class we can create "void figure () method" instead of "void castle () method", and overwrite this method in all sub classes, and from the reference the method of the object will be invoked.
3. The "void figure () method" In Figure class we can convert to abstract, because the object Figure we don't need to invoke this method, only for the subclasses.

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

While we travel through the list of objects, we can ask if the object belongs to a certain class, using the reserved word “instance of”.

For example, in this code we asked if figure is a Queen.

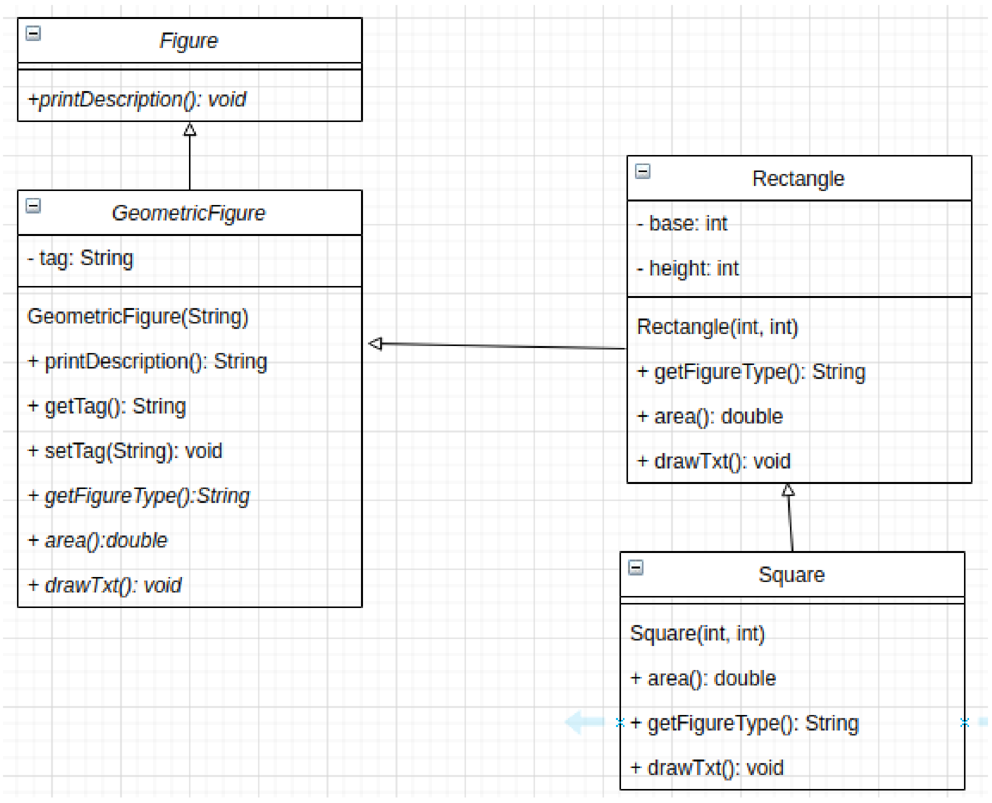


```
private int getCantQueens() {
    int queens = 0;
    for (Figure figure: figures) {
        if ( figure instanceof Queen) {
            queens ++;
        }
    }
    return queens;
}
```

****

Exercise 2 to practice with Figures

Class diagram



5. Answer the following questions:

o Show the difference between a class and an object.

A class is a template what generalize the features a set objects and an object is an instance of class; the objects are an entity at runtime.

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

The steps for a creation of objects are:

1. **Declaration:** a variable of an object is declared: example “Figure figure = ”.
2. **Instantiation:** the “new” keyword is used to create an object. Example “Figure figure= new”
3. **Initialization:** The "new' keyword is followed by a call a constructor. This call initializes the new object. Example “Figure figure = new Figure(); “.

o How is an object instantiated in Java?

In java first a variable is created then the word new is used then we call the constructor of the class. Example

Figure figure = new Figure();

Answer the following questions:

o What is inheritance?

Inheritance is a mechanism in which one object acquires all the properties and behaviors of a parent object, in java is used for code reuse.

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

In order to inherit the attributes and methods of the super class, the reserved word "extends" is placed in the sub class, then to access the constructor of the superclass, we put “super” in the constructor of the sub class.

o Which methods of the superclass are visible from the subclasses?

For the subclass all the methods of the superclass are visible, except the abstract methods, this are implemented in sub classes.

o What is the meaning of method overriding?

Meaning that allows a subclass to provide a specific implementation of a method that is already provided by one of its super class.

● Which type are both instantiated objects (in options 1 and 2)?

Are instanced of type Rectangle and Square respectively.

● Which type is the variable that references them?

Are referenced from the GeometricFigure type variable.

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

By inheritance all the methods of the superclass are seen from the subclasses

● **Can you use the same variable as a reference for different types of figures? Why?**

if more geometric figures are added, if I could use the same reference, because everyone is going to inherit from the super class GeometricFigure.