Sahil

**Encapsulation:**

In our program, we use encapsulation in all of our classes and subclasses that feed into main. For example, in BoardGUI, we created accessors for various items, including the window which we then used in main. Piece and Square also have assessors, and you would only need to know the name of the assessor in order to use it. We were able to keep encapsulation by not referencing variables from other classes inside of main directly For example, instead of using piece.color in main, with .color being an instance variable of piece, we used piece.checkColor() . In order for the user to call piece.color, they would have to know what is happening inside of Piece, and would break encapsulation.

**Polymorphism:**

We used polymorphism in our subclasses of Piece. The only method we had in the subclasses was a method called ‘calcListDirections()’. By keeping the name constant in all the subclasses, it made it much easier to code since we wouldn’t have to worry about whether or not the each piece object had calcListDirections as a method in the subclass. We were able to have shorter and code that was less prone to bugs.

**Inheritance:**

We used inheritance in our king, bishop, rook etc. subclasses. We did this since each piece would be undergoing very similar calculations, so we decided to create a Piece superclass. We also used inheritance for Square, with Square being a subclass of Button. We had to overwrite the init because we had some slight differences (ex: setting the color depending on the coordinates, saving the lower left and upper right corners as instance variables). We were able to reuse some of the methods from Button, like the .clicked(pt) method.