1. Try the codes in the previous slides, what can you say and how to improve?

```
abstract class Pen {
   protected String engraved;
   protected String inkColor;

   public abstract void showProperty();
}

class Pencil extends Pen {
   private String brand;
   private int number;
   private String status;

   public Pencil(String brand, int number, String status) {
      this.brand = brand;
      // Adding validation for number
      if (number <= 0) {
         throw new IllegalArgumentException("Number must be positive.");
      }
      this.number = number;
      this.status = status;
   }

   @Override
   public void showProperty() {
      System.out.println("This pencil is a " + brand + " pencil, number " + number + ".");
      System.out.println("Status: " + status);
 }
}

public class Main {
   public static void main(String[] args) {
      Pen myPencil = new Pencil("Faber-Castell", 2, "sharpened");
      myPencil.showProperty();
   }
}
```

In this code the improvement that needs are:

: While the properties in the Pen class are currently private, making them protected would allow sub classes like Pencil to access them directly without the need for getter methods.
Error Handling: Currently, there's no error handling for invalid inputs. You could add validation checks in the constructor to ensure that only valid values are accepted.
Documentation: Adding comments to your code can improve readability and make it easier for others to understand its purpose and functionality.
Naming Conventions: While your variable names are clear and descriptive, following Java naming conventions more strictly would involve starting class names with an uppercase letter (Pen instead of pen).