Name: Ethan Baraf

AI Reflection Document

Creating effective prompts that direct AI to generate code can be difficult. Companies have coined a new phrase to describe this: *prompt engineering.*

Record your experience using AI to generate the Sprite inheritance hierarchy for your board game.

Copy and paste your whole AI conversation here. Include your prompts and the AI responses (this might be pretty long, but that is okay).

please make a java abstract class called Sprite. please leave it empty for now.

I'll create a simple Java abstract class called Sprite for you.

Java

```java
/**
 * Abstract class representing a Sprite object.
 */
public abstract class Sprite {
    // Empty for now as requested
}
```

I've created an empty Java abstract class called Sprite as requested. The class is completely empty with just the abstract class declaration.

Would you like me to add any fields, constructors, or abstract methods to this class?

please import JPanel Graphics Color and Dimension

I'll update the Sprite class to include the imports for JPanel, Graphics, Color, and Dimension.

Java

```java
import javax.swing.JPanel;
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Dimension;

/**
 * Abstract class representing a Sprite object.
 */
public abstract class Sprite {
    // Empty for now as requested
}
```

I've updated the Sprite class to include the requested imports:

Java

```
javax.swing.JPanel
java.awt.Graphics
java.awt.Color
java.awt.Dimension
```

The class body remains empty as previously requested. Would you like me to add any fields, methods, or other elements to the Sprite class now?

now please make an abstract method called DrawMe that takes in Graphics g, int x, int y, int offsetX, int offsetY

I'll add the abstract method DrawMe with the specified parameters to the Sprite class.

Java

```java
import javax.swing.JPanel;
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Dimension;

/**
 * Abstract class representing a Sprite object.
 */
public abstract class Sprite {
    /**
     * Abstract method to draw the sprite.
     *
     * @param g The Graphics object to draw on
     * @param x The x coordinate for drawing
     * @param y The y coordinate for drawing
     * @param offsetX The x offset
     * @param offsetY The y offset
     */
    public abstract void DrawMe(Graphics g, int x, int y, int offsetX, int
offsetY);
}
```

I've added the abstract method DrawMe to the Sprite class with the parameters you specified:

Java

```java
public abstract void DrawMe(Graphics g, int x, int y, int
offsetX, int offsetY);
```

The method includes JavaDoc comments that explain each parameter:

Java

```java
g: The Graphics object to draw on
x: The x coordinate for drawing
y: The y coordinate for drawing
offsetX: The x offset
offsetY: The y offset
```

Since this is an abstract method, it has no implementation and will need to be implemented by any concrete subclasses of Sprite.
Would you like me to add any additional methods or fields to the Sprite class?

please add a private Color variable called color that is a new color with an rgb of 230, 255, 255
I'll add the private Color variable with the specified RGB values to the Sprite class.

Java

```java
import javax.swing.JPanel;
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Dimension;

/**
 * Abstract class representing a Sprite object.
 */
public abstract class Sprite {
    // Private color variable with RGB values (230, 255, 255) - light cyan
    private Color color = new Color(230, 255, 255);

    /**
     * Abstract method to draw the sprite.
     *
     * @param g The Graphics object to draw on
     * @param x The x coordinate for drawing
     * @param y The y coordinate for drawing
     * @param offsetX The x offset
     * @param offsetY The y offset
     */
    public abstract void DrawMe(Graphics g, int x, int y, int offsetX, int
offsetY);
}
```

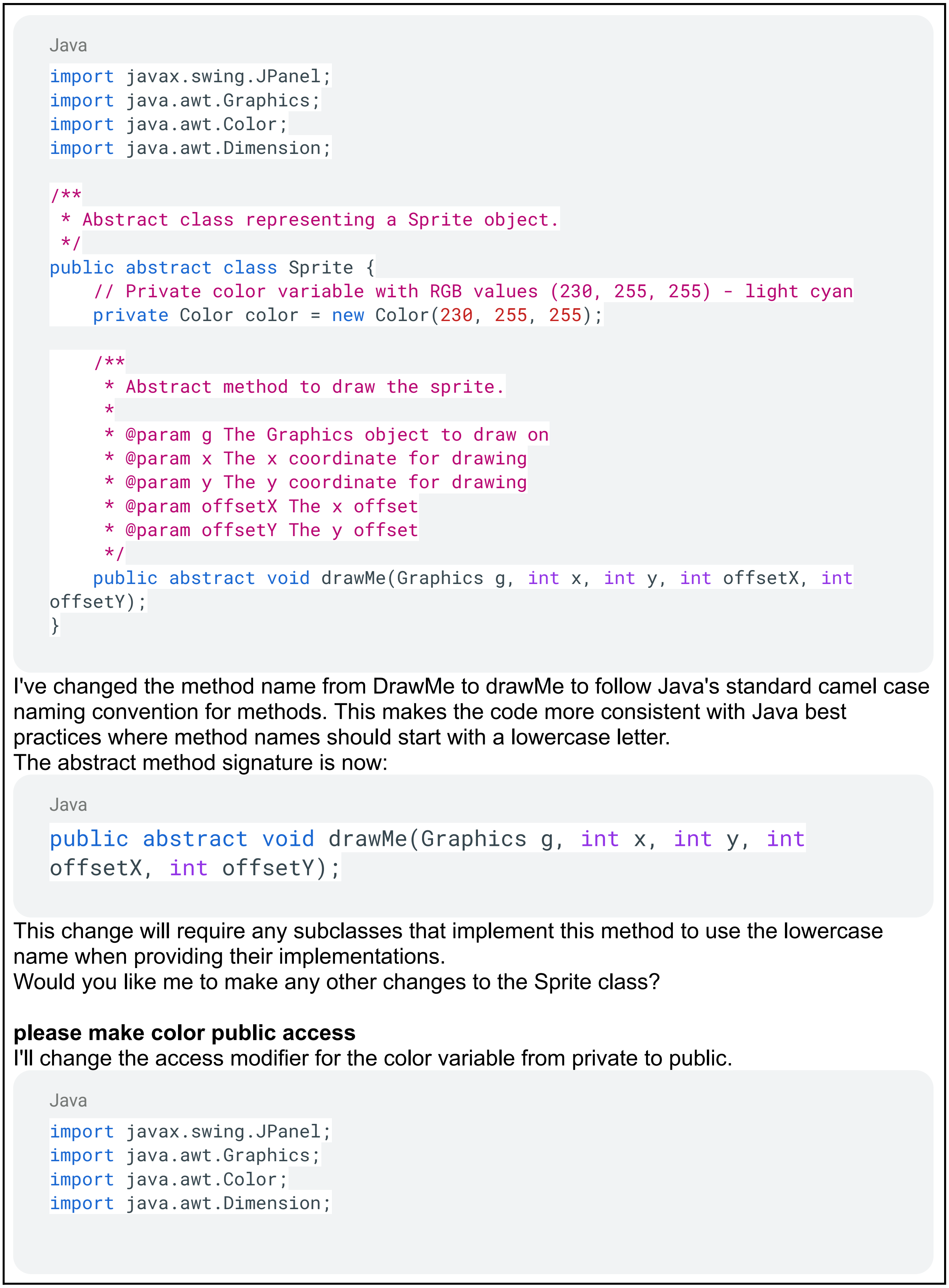
I've added the private Color variable called color with RGB values of (230, 255, 255), which creates a light cyan color:

Java

```java
private Color color = new Color(230, 255, 255);
```

The variable is now part of the Sprite class, along with the abstract DrawMe method. Any subclasses of Sprite will inherit this color property, though they would need getter and setter methods to access or modify it since it's private.
Would you like me to add any getter or setter methods for the color variable, or make any other changes to the class?

please change the naming of the method to drawMe
I'll update the method name from DrawMe to drawMe to follow Java's standard camel case naming convention for methods.



Java

```java
import javax.swing.JPanel;
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Dimension;

/**
 * Abstract class representing a Sprite object.
 */
public abstract class Sprite {
    // Private color variable with RGB values (230, 255, 255) - light cyan
    private Color color = new Color(230, 255, 255);

    /**
     * Abstract method to draw the sprite.
     *
     * @param g The Graphics object to draw on
     * @param x The x coordinate for drawing
     * @param y The y coordinate for drawing
     * @param offsetX The x offset
     * @param offsetY The y offset
     */
    public abstract void drawMe(Graphics g, int x, int y, int offsetX, int
offsetY);
}
```

I've changed the method name from DrawMe to drawMe to follow Java's standard camel case naming convention for methods. This makes the code more consistent with Java best practices where method names should start with a lowercase letter.

The abstract method signature is now:

Java

```java
public abstract void drawMe(Graphics g, int x, int y, int
offsetX, int offsetY);
```

This change will require any subclasses that implement this method to use the lowercase name when providing their implementations.

Would you like me to make any other changes to the Sprite class?

please make color public access

I'll change the access modifier for the color variable from private to public.

Java

```java
import javax.swing.JPanel;
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Dimension;
```

```
/**
 * Abstract class representing a Sprite object.
 */
public abstract class Sprite {
    // Public color variable with RGB values (230, 255, 255) - light cyan
    public Color color = new Color(230, 255, 255);

    /**
     * Abstract method to draw the sprite.
     *
     * @param g The Graphics object to draw on
     * @param x The x coordinate for drawing
     * @param y The y coordinate for drawing
     * @param offsetX The x offset
     * @param offsetY The y offset
     */
    public abstract void drawMe(Graphics g, int x, int y, int offsetX, int
offsetY);
}
```

I've changed the access modifier for the color variable from private to public. Now the color can be directly accessed and modified by any subclasses of Sprite and by any other classes.

Java

```java
public Color color = new Color(230, 255, 255);
```

I also updated the comment to reflect that this is now a public field. With this change, external code will be able to both read and modify the color directly without needing getter or setter methods.

Would you like me to make any other changes to the Sprite class?

What worked well when using AI to generate your inheritance hierarchy? What were some challenges when using AI to generate your inheritance hierarchy?

Considering i already have so much set up, i tried to figure out the best way to put the class to use, which would mean just making a variable and method. In this instance, i found that having it do very minimal ensured that i got what i wanted out of it. I made sure to ask for very small things as i went.

What computer science concepts do you need to know in order to complete this task and integrate it successfully into your code?

At the end of the day, i found that this required a pretty hefty amount of background knowledge, but i made it work out. Previous attempts at using ai for personal projects proved to be far less useful than just learning how to code on my own, so i have been trying to work on ensuring that i dont use ai for any coding.