Saron Berhane
Week 5 - Design Patterns

1. Behavioral Pattern: Observer Pattern

The Observer pattern allows us to notify one or more objects when the state of a particular object changes. For our blog system, we can use the Observer pattern to notify subscribers when a new blog post is added.

```
class Blog {
  constructor() {
    this.subscribers = [];
  }

  subscribe(observer) {
    this.subscribers.push(observer);
  }

  unsubscribe(observer) {
    this.subscribers = this.subscribers.filter((sub) => sub !==
observer);
  }

  addPost(post) {
    // add post to the database
    this.notifySubscribers(post);
  }

  notifySubscribers(post) {
    this.subscribers.forEach((sub) => sub.notify(post));
  }
}

class Subscriber {
  notify(post) {
    console.log(`New blog post: ${post.title}`);
  }
}

const blog = new Blog();
const subscriber1 = new Subscriber();
const subscriber2 = new Subscriber();
blog.subscribe(subscriber1);
blog.subscribe(subscriber2);
blog.addPost({ title: 'New blog post', content: 'Lorem ipsum...' });
```

2. Creational Pattern: Singleton Pattern

The Singleton pattern ensures that there is only one instance of a particular class. For our blog system, we can use the Singleton pattern to ensure that we have only one instance of the database connection, which can be used throughout the application.

```
class DatabaseConnection {
  constructor() {
    this.connection = null;
  }

  static getInstance() {
    if (!this.connection) {
      this.connection = new DatabaseConnection();
    }
    return this.connection;
  }

  connect() {
    // connect to the database
  }

  disconnect() {
    // disconnect from the database
  }
}

const dbConnection = DatabaseConnection.getInstance();
dbConnection.connect();
```

3. Structural Pattern: Facade Pattern

The Facade pattern provides a simplified interface to a complex system. For our blog system, we can use the Facade pattern to provide a simplified interface to the database, making it easier to interact with.

```
class DatabaseFacade {
  constructor() {
    this.connection = null;
  }

  connect() {
    if (!this.connection) {
      this.connection = new DatabaseConnection();
      this.connection.connect();
    }
  }

  disconnect() {
    if (this.connection) {
      this.connection.disconnect();
      this.connection = null;
    }
  }

  addPost(post) {
    // add post to the database
  }

  getPosts() {
    // get all posts from the database
  }
}

const dbFacade = new DatabaseFacade();
dbFacade.connect();
dbFacade.addPost({ title: 'New blog post', content: 'Lorem
ipsum...' });
```