Problema:
Você é um desenvolvedor de jogos e está criando um novo jogo de plataforma chamado "Jumpin' Jack". Neste jogo, o personagem principal, Jack, pode pular em diferentes plataformas que estão localizadas em diferentes alturas. Além disso, Jack pode pular em diferentes tipos de plataformas, como madeira, pedra e gel. Cada tipo de plataforma tem um comportamento diferente quando Jack pula nela.

Por exemplo, se Jack pular em uma plataforma de madeira, ele pode pular normalmente. Mas se Jack pular em uma plataforma de gelo, ele pode deslizar e perder o controle por um breve período de tempo. Além disso, se Jack pular em uma plataforma de pedra, ele pode pular mais alto do que o normal.

Como você pode garantir que cada tipo de plataforma tenha seu comportamento único quando Jack pula nela, mas ainda assim manter o mesmo fluxo geral do jogo?

Solução:
Podemos usar o padrão de projeto Template Method para resolver esse problema. Neste caso, podemos criar uma classe abstrata chamada Platform que define o fluxo geral do jogo quando Jack pula em qualquer plataforma. Uma classe Platform terá um método abstrato chamado jump() que será implementado por cada tipo de plataforma.
Aqui está a implementação da classe Platform:

```java
public abstract class Platform {
    public void jump() {
        beforeJump();
        jumpBehavior();
        afterJump();
    }

    protected abstract void jumpBehavior();

    private void beforeJump() {
        // código comum antes de Jack pular
    }

    private void afterJump() {
        // código comum depois de Jack pular
    }
}
```

Uma classe Platform tem um método jump() que é o método do Modelo. Ele chama os métodos beforeJump() e afterJump() que contêm o código comum antes e depois de Jack pular, respeitosamente. O método jumpBehavior() é um método abstrato que será implementado por cada tipo de plataforma.

Agora, vamos criar como classes concretas para cada tipo de plataforma.

Aqui está a implementação da classe WoodPlatform:

```java
public class WoodPlatform extends Platform {
    @Override
    protected void jumpBehavior() {
        // código para Jack pular em uma plataforma de madeira
    }
}
```

Aqui está a implementação da classe StonePlatform:

```java
public class StonePlatform extends Platform {
    @Override
    protected void jumpBehavior() {
        // código para Jack pular em uma plataforma de pedra
    }
}
```

Aqui está a implementação da classe IcePlatform:

Dessa forma, cada tipo de plataforma tem seu comportamento único quando Jack pula nela, mas ainda assim mantém o mesmo fluxo geral do jogo. Além disso, se precisarmos adicionar um novo tipo de plataforma no futuro, basta criar uma nova classe que estenda a classe Platform e implementar o método jumpBehavior().

java

```java
public class IcePlatform extends Platform {
    @Override
    protected void jumpBehavior() {
        // código para Jack pular em uma plataforma de gelo
    }
}
```