CS 351 Design of Large Programs Template Method Pattern

Brooke Chenoweth

University of New Mexico

Spring 2025

Human vs Computer Game

- All players play by the same rules
- Human turn needs input from user
- Computer somehow chooses move
- Legality, scoring, updates same for both

Console or GUI Game

- Game logic is the same
- Console game displays text
- GUI game updates graphical display

Preparing Hot Beverages

Tea

- 1. Boil some water
- 2. Steep tea in the water
- 3. Pour into cup
- 4. Add lemon

Preparing Hot Beverages

Tea

- 1. Boil some water
- 2. Steep tea in the water
- 3. Pour into cup
- 4. Add lemon

Coffee

- 1. Boil some water
- 2. Brew coffee with the water
- 3. Pour into cup
- 4. Add cream and sugar

Preparing Hot Beverages

Tea

- 1. Boil some water
- 2. Steep tea in the water
- 3. Pour into cup
- 4. Add lemon

Coffee

- 1. Boil some water
- 2. Brew coffee with the water
- 3. Pour into cup
- 4. Add cream and sugar

Steps 1 and 3 are the same Steps 2 and 4 differ, but serve similar purpose

Hot Beverages in Java

```
public abstract class HotDrink {
  public void prepareDrink() {
    boilWater();
    brew();
    pourIntoCup();
    addCondiments();
  }
  protected void boilWater() {
    System.out.println("Boiling water");
  }
  protected void pourIntoCup() {
    System.out.println("Pouring into cup");
  }
  protected abstract void brew();
  protected abstract void addCondiments();
}
```

Hot Beverages in Java

```
public class Tea extends HotDrink {
    protected void brew() {
        System.out.println("Steeping the tea");
    }
    protected void addCondiments() {
        System.out.println("Adding lemon");
    }
}
```

```
public class Coffee extends HotDrink {
   protected void brew() {
      System.out.println("Pouring water over grounds");
   }
   protected void addCondiments() {
      System.out.println("Adding cream and sugar");
   }
}
```

The Template Method Pattern

The *Template Method Pattern* defines the skeleton of an algorithm in a method, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

Template Method

- Template method in the abstract parent class defines algorithm as sequence of steps.
- Can make template method final to stop subclasses from changing the steps.
- Some steps may be concrete methods defined in the parent class.
- Some steps are abstract and must be implemented in the subclasses.
- Some steps may be optional, so parent provides a *hook* for the subclass to maybe override.

Hooks

- A *hook* is a method used in the template method that has a concrete implementation that does nothing. (Or some other simple default)
- Subclasses may override to actually do something, but don't have to.

Hot Beverage with a Hook

```
public abstract class HotDrinkWithHook {
  public void prepareDrink() {
    boilWater();
    brew();
    pourIntoCup();
    addCondiments();
  }
  // boilWater, pourIntoCup
  protected abstract void brew();
  protected void addCondiments() { }
}
```

```
public class BlackCoffee extends HotDrinkWithHook {
   protected void brew() {
      System.out.println("Pouring water over grounds");
   }
}
```