**Author:** Pedro Jahir Hinojosa García

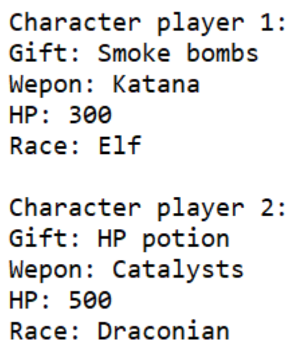
**Instructor:** Miguel Rugerio

**Activity:** Builder method

**Date:** 31/08/2024

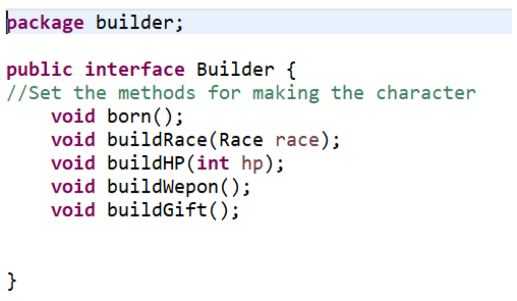
The main purpose of this method is to create objects with similar dependencies but different characteristics. In this case, I implemented a character builder that assigns the same attributes to different characters while allowing changes to the internal data without the need to override methods each time a character is built. An example of this implementation is shown in Image 1.

First, I added the builder class as shown in Code 1, where I declare all the methods that my characters will have. To define the attributes, each character class must implement the builder interface to override the methods, as shown in Code 2. Then, I added the constructor and the toString method, as seen in Code 3. After that, I set the specific attributes in the Director class, shown in Code 4. Finally, in the main class, I create a new Director object to assign the attributes to the characters, as demonstrated in Code 5.



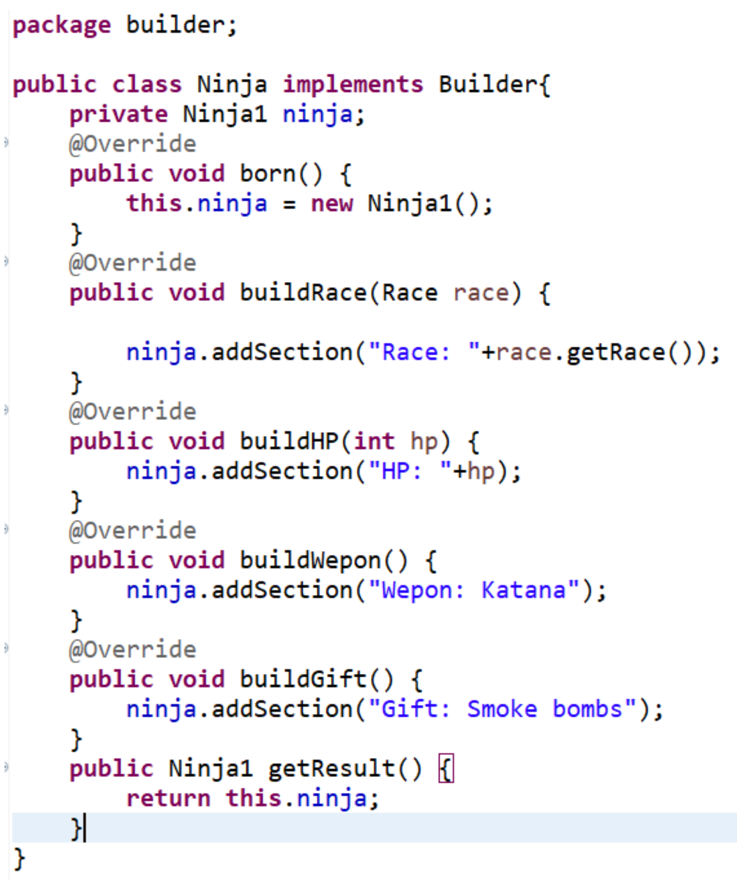
```
Character player 1:
Gift: Smoke bombs
Wepon: Katana
HP: 300
Race: Elf

Character player 2:
Gift: HP potion
Wepon: Catalysts
HP: 500
Race: Draconian
```

Image 1: Two character build with builder method.



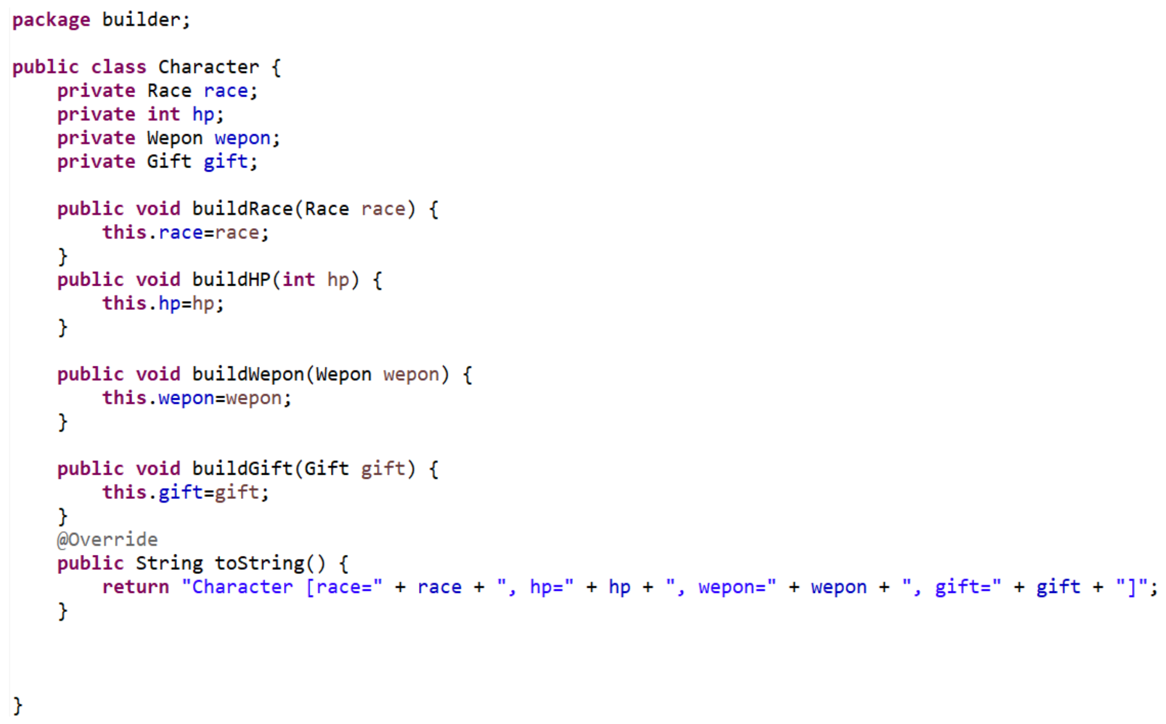
```java
package builder;

public interface Builder {
//Set the methods for making the character
    void born();
    void buildRace(Race race);
    void buildHP(int hp);
    void buildWepon();
    void buildGift();


}
```

Code 1: The builder.



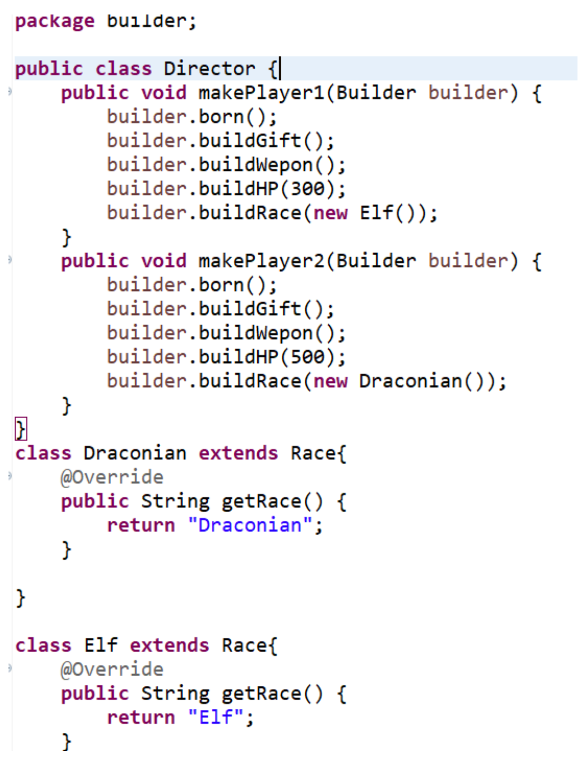
```java
package builder;

public class Ninja implements Builder{
    private Ninja1 ninja;
    @Override
    public void born() {
        this.ninja = new Ninja1();
    }
    @Override
    public void buildRace(Race race) {

        ninja.addSection("Race: "+race.getRace());
    }
    @Override
    public void buildHP(int hp) {
        ninja.addSection("HP: "+hp);
    }
    @Override
    public void buildWepon() {
        ninja.addSection("Wepon: Katana");
    }
    @Override
    public void buildGift() {
        ninja.addSection("Gift: Smoke bombs");
    }
    public Ninja1 getResult() {
        return this.ninja;
    }
}
```

Code 2: A class that implements Builder.



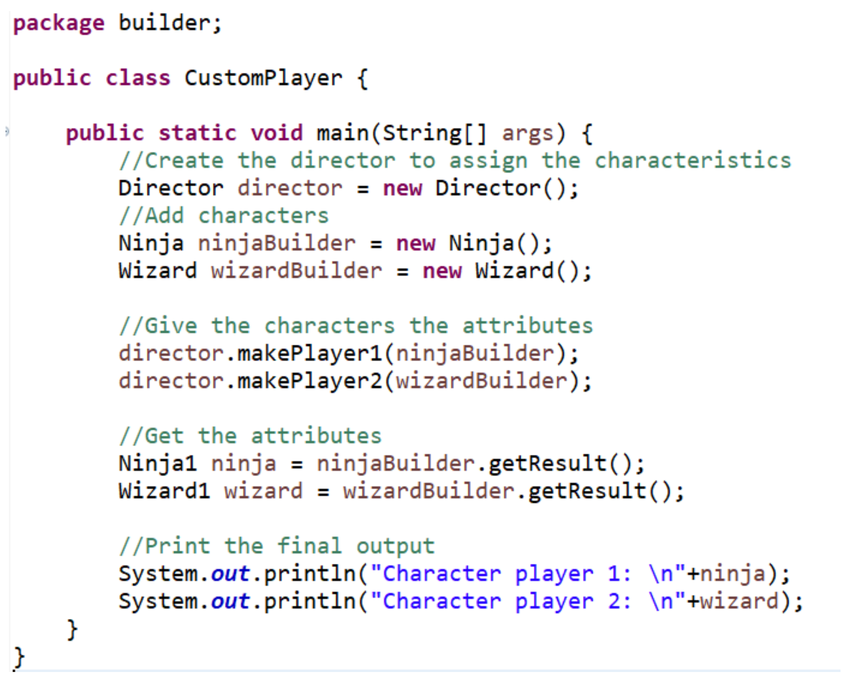
```java
package builder;

public class Character {
    private Race race;
    private int hp;
    private Wepon wepon;
    private Gift gift;

    public void buildRace(Race race) {
        this.race=race;
    }
    public void buildHP(int hp) {
        this.hp=hp;
    }

    public void buildWepon(Wepon wepon) {
        this.wepon=wepon;
    }

    public void buildGift(Gift gift) {
        this.gift=gift;
    }
    @Override
    public String toString() {
        return "Character [race=" + race + ", hp=" + hp + ", wepon=" + wepon + ", gift=" + gift + "]";
    }



}
```

Code 3: Character class declaring the attributes and the toString method.



```java
package builder;

public class Director {
    public void makePlayer1(Builder builder) {
        builder.born();
        builder.buildGift();
        builder.buildWepon();
        builder.buildHP(300);
        builder.buildRace(new Elf());
    }
    public void makePlayer2(Builder builder) {
        builder.born();
        builder.buildGift();
        builder.buildWepon();
        builder.buildHP(500);
        builder.buildRace(new Draconian());
    }
}
class Draconian extends Race{
    @Override
    public String getRace() {
        return "Draconian";
    }

}

class Elf extends Race{
    @Override
    public String getRace() {
        return "Elf";
    }
```

Code 4: Director class assigning specific characteristics.



```java
package builder;

public class CustomPlayer {

    public static void main(String[] args) {
        //Create the director to assign the characteristics
        Director director = new Director();
        //Add characters
        Ninja ninjaBuilder = new Ninja();
        Wizard wizardBuilder = new Wizard();

        //Give the characters the attributes
        director.makePlayer1(ninjaBuilder);
        director.makePlayer2(wizardBuilder);

        //Get the attributes
        Ninja1 ninja = ninjaBuilder.getResult();
        Wizard1 wizard = wizardBuilder.getResult();

        //Print the final output
        System.out.println("Character player 1: \n"+ninja);
        System.out.println("Character player 2: \n"+wizard);
    }
}
```

Code 5: Main class that join all the characteristics set by the director.