Design Pattern - Builder

Gabriel Colling,
José Augusto Accorsi e
Pedro Bohlmann Cascaes Silva

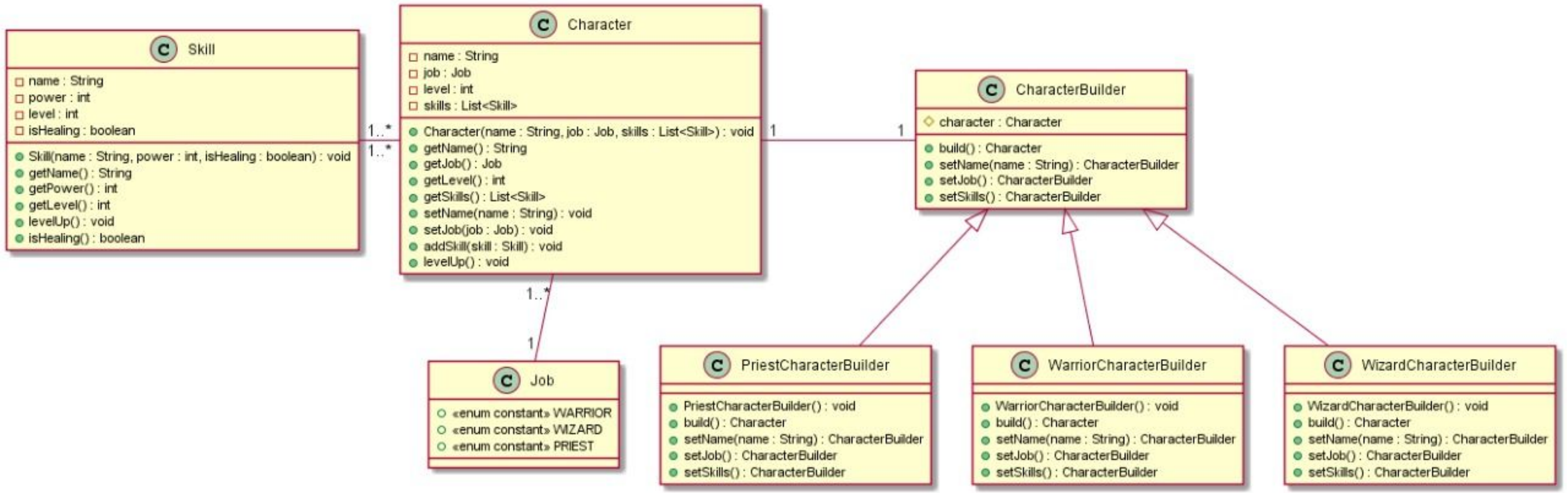
Objetivo do padrão

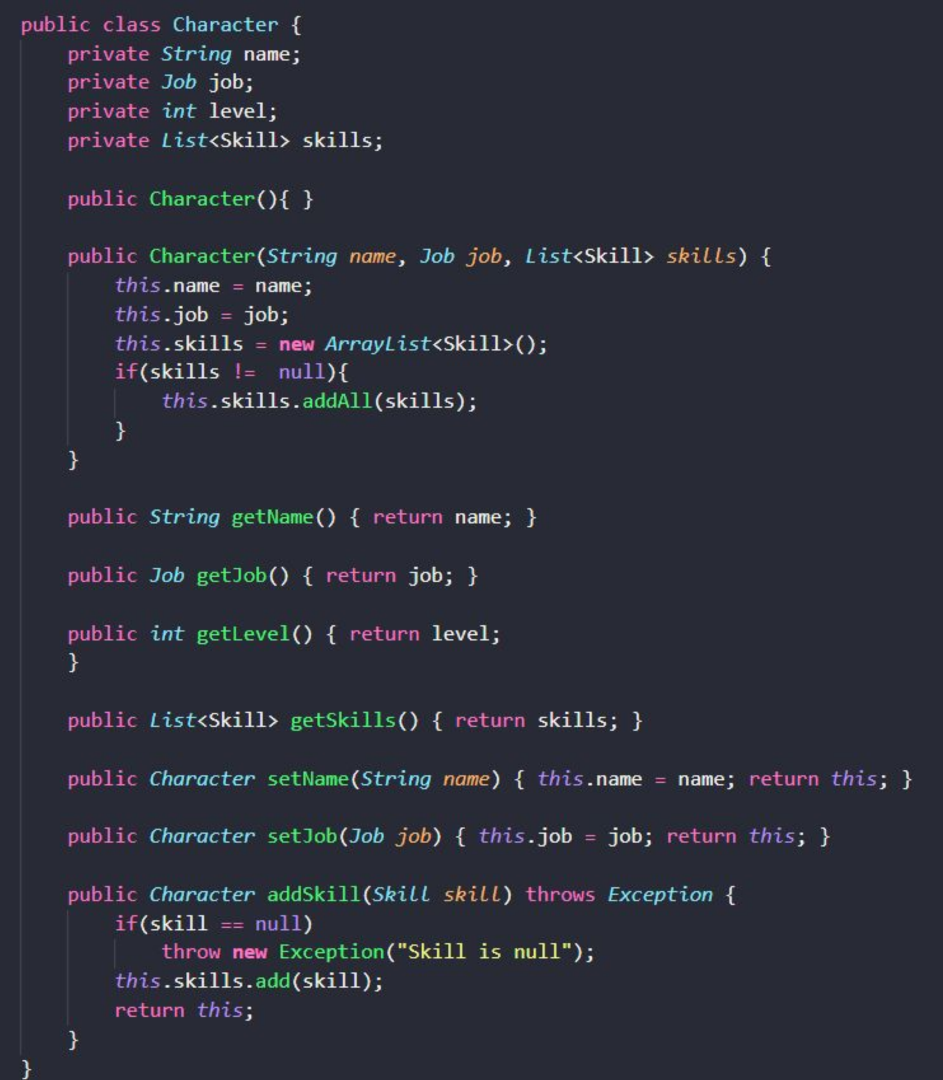
Facilitar a construção de um objeto complexo onde já existe um especificação definida.

Solução proposta

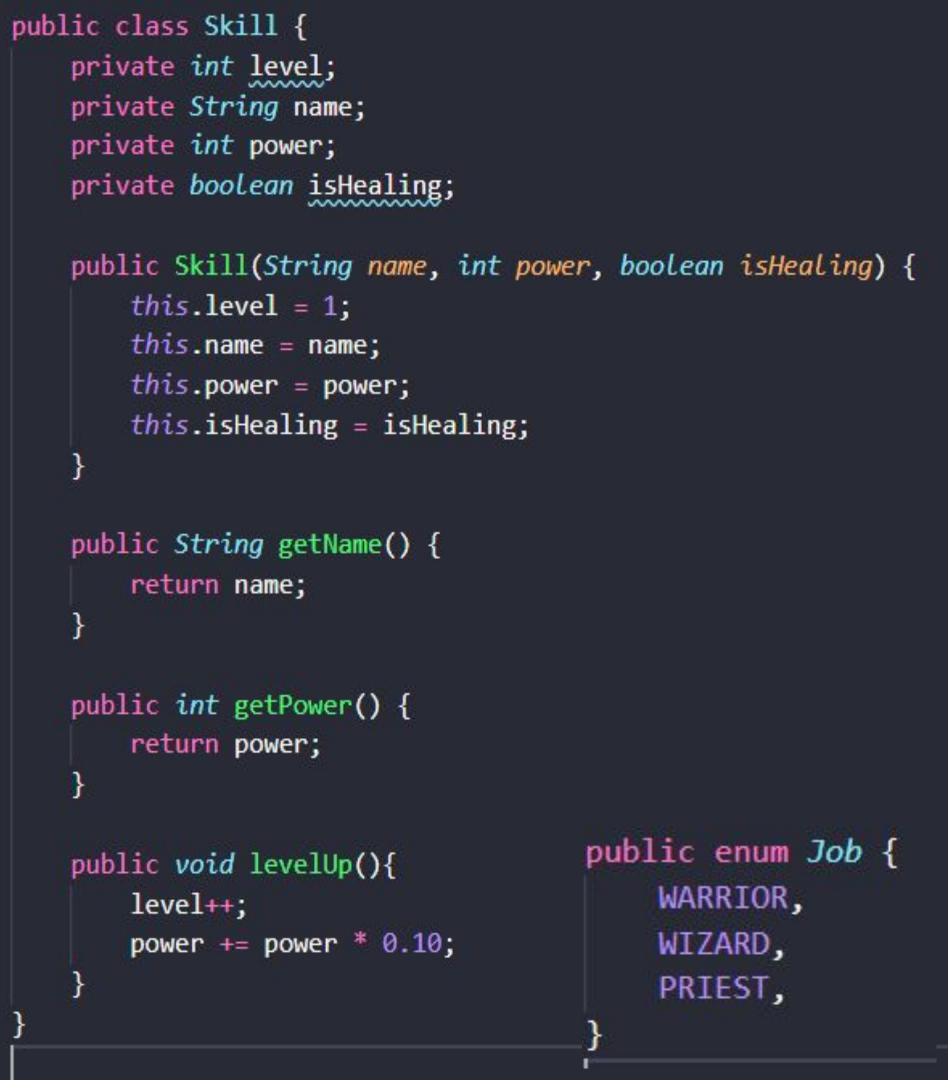
A partir da classe base de dados criar um uma classe auxiliar que tem como objetivo montar os dados necessários para gerar uma instância utilizando os providos anteriormente.

Diagrama de classe do exemplo


Skill
name : String
power : int
level : int
isHealing : boolean
Skill(name : String, power : int, isHealing : boolean) : void
getName() : String
getPower() : int
getLevel() : int
levelUp() : void
isHealing() : boolean
1..*
1..*
Character
name : String
job : Job
level : int
skills : List<Skill>
Character(name : String, job : Job, skills : List<Skill>) : void
getName() : String
getJob() : Job
getLevel() : int
getSkills() : List<Skill>
setName(name : String) : void
setJob(job : Job) : void
addSkill(skill : Skill) : void
levelUp() : void
1
1
CharacterBuilder
character : Character
build() : Character
setName(name : String) : CharacterBuilder
setJob() : CharacterBuilder
setSkills() : CharacterBuilder
1..*
1
Job
«enum constant» WARRIOR
«enum constant» WIZARD
«enum constant» PRIEST
PriestCharacterBuilder
PriestCharacterBuilder() : void
build() : Character
setName(name : String) : CharacterBuilder
setJob() : CharacterBuilder
setSkills() : CharacterBuilder
WarriorCharacterBuilder
WarriorCharacterBuilder() : void
build() : Character
setName(name : String) : CharacterBuilder
setJob() : CharacterBuilder
setSkills() : CharacterBuilder
WizardCharacterBuilder
WizardCharacterBuilder() : void
build() : Character
setName(name : String) : CharacterBuilder
setJob() : CharacterBuilder
setSkills() : CharacterBuilder

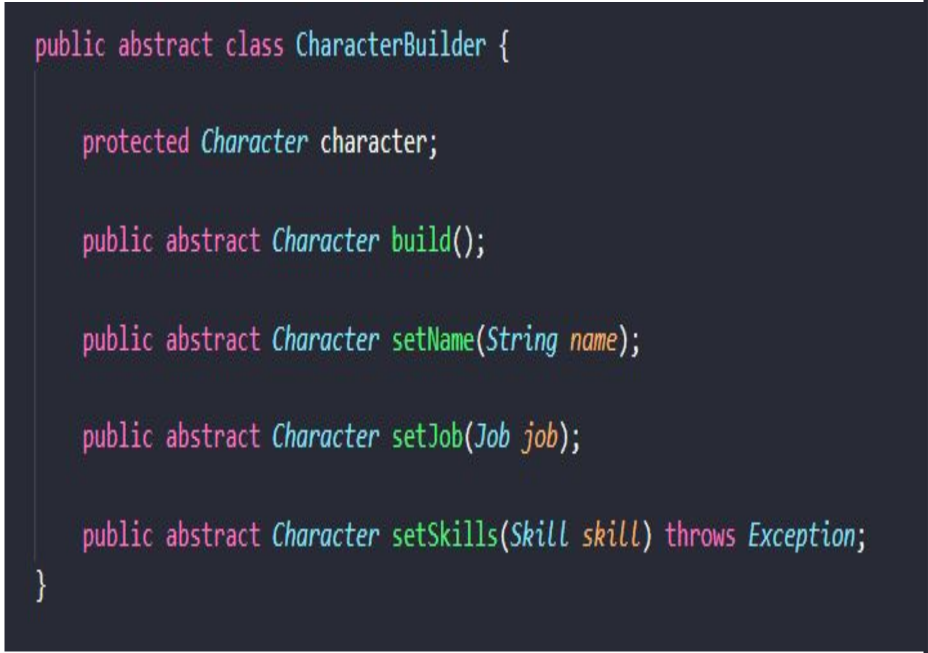


```java
public class Character {
    private String name;
    private Job job;
    private int level;
    private List<Skill> skills;

    public Character(){ }

    public Character(String name, Job job, List<Skill> skills) {
        this.name = name;
        this.job = job;
        this.skills = new ArrayList<Skill>();
        if(skills !=  null){
            this.skills.addAll(skills);
        }
    }

    public String getName() { return name; }

    public Job getJob() { return job; }

    public int getLevel() { return level;
    }

    public List<Skill> getSkills() { return skills; }

    public Character setName(String name) { this.name = name; return this; }

    public Character setJob(Job job) { this.job = job; return this; }

    public Character addSkill(Skill skill) throws Exception {
        if(skill == null)
            throw new Exception("Skill is null");
        this.skills.add(skill);
        return this;
    }
}
```

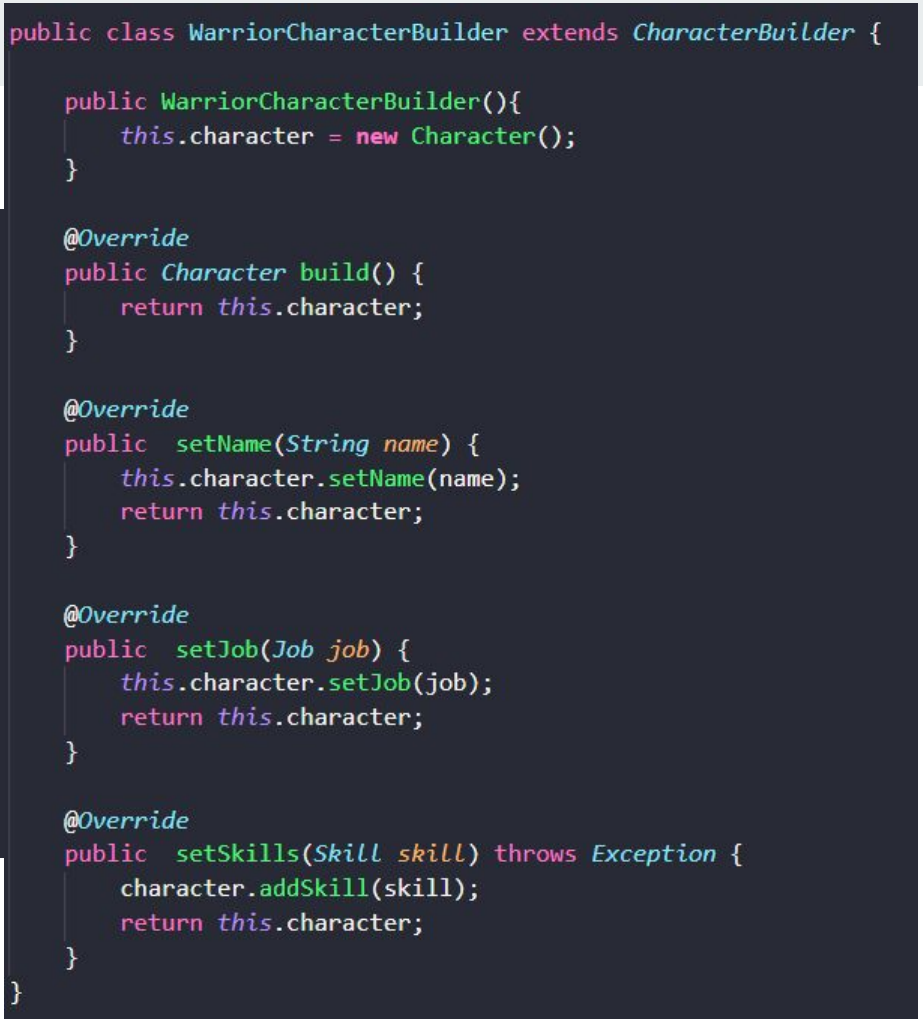


```java
public class Skill {
    private int level;
    private String name;
    private int power;
    private boolean isHealing;

    public Skill(String name, int power, boolean isHealing) {
        this.level = 1;
        this.name = name;
        this.power = power;
        this.isHealing = isHealing;
    }

    public String getName() {
        return name;
    }

    public int getPower() {
        return power;
    }

    public void levelUp(){
        level++;
        power += power * 0.10;
    }
}
```

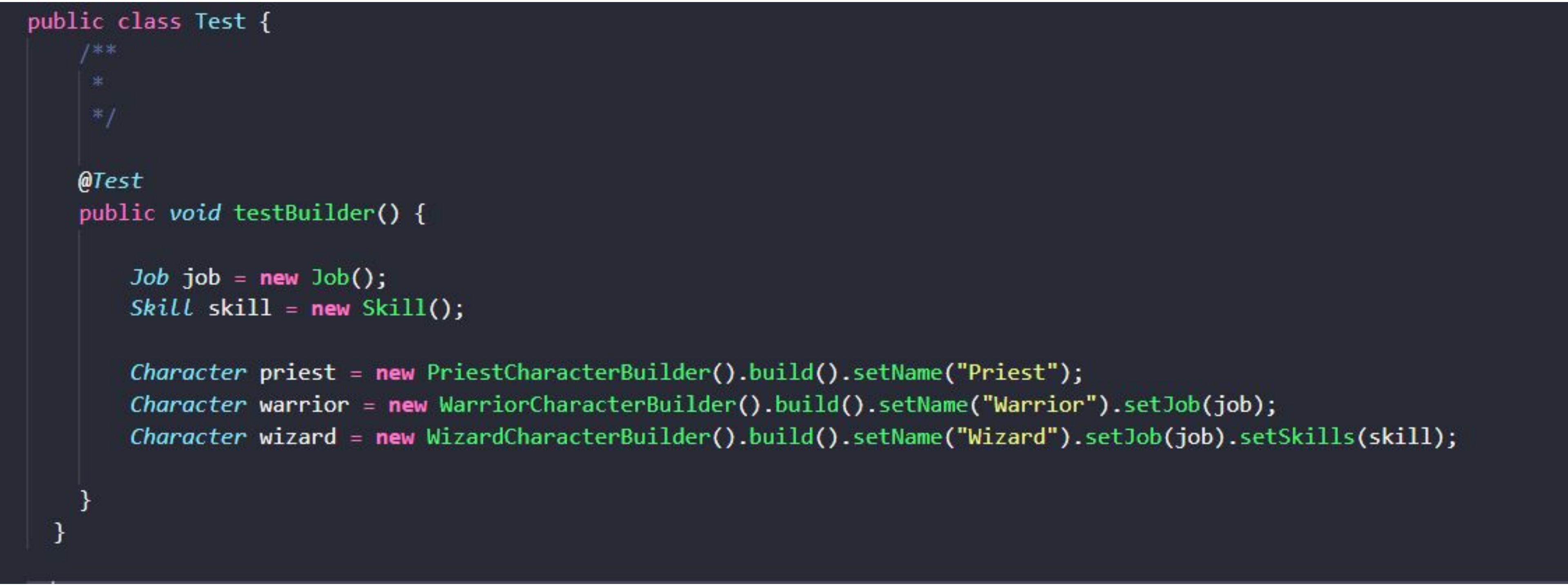
```java
public enum Job {
    WARRIOR,
    WIZARD,
    PRIEST,
}
```



```java
public abstract class CharacterBuilder {

    protected Character character;

    public abstract Character build();

    public abstract Character setName(String name);

    public abstract Character setJob(Job job);

    public abstract Character setSkills(Skill skill) throws Exception;
}
```



```java
public class WarriorCharacterBuilder extends CharacterBuilder {

    public WarriorCharacterBuilder(){
        this.character = new Character();
    }

    @Override
    public Character build() {
        return this.character;
    }

    @Override
    public  setName(String name) {
        this.character.setName(name);
        return this.character;
    }

    @Override
    public  setJob(Job job) {
        this.character.setJob(job);
        return this.character;
    }

    @Override
    public  setSkills(Skill skill) throws Exception {
        character.addSkill(skill);
        return this.character;
    }
}
```



```java
public class Test {
    /**
     *
     */

    @Test
    public void testBuilder() {

        Job job = new Job();
        Skill skill = new Skill();

        Character priest = new PriestCharacterBuilder().build().setName("Priest");
        Character warrior = new WarriorCharacterBuilder().build().setName("Warrior").setJob(job);
        Character wizard = new WizardCharacterBuilder().build().setName("Wizard").setJob(job).setSkills(skill);

    }
}
```