Design Pattern - Builder

Gabriel Colling,
José Augusto Accorsi e
Pedro Bohlmann Cascaes Silva

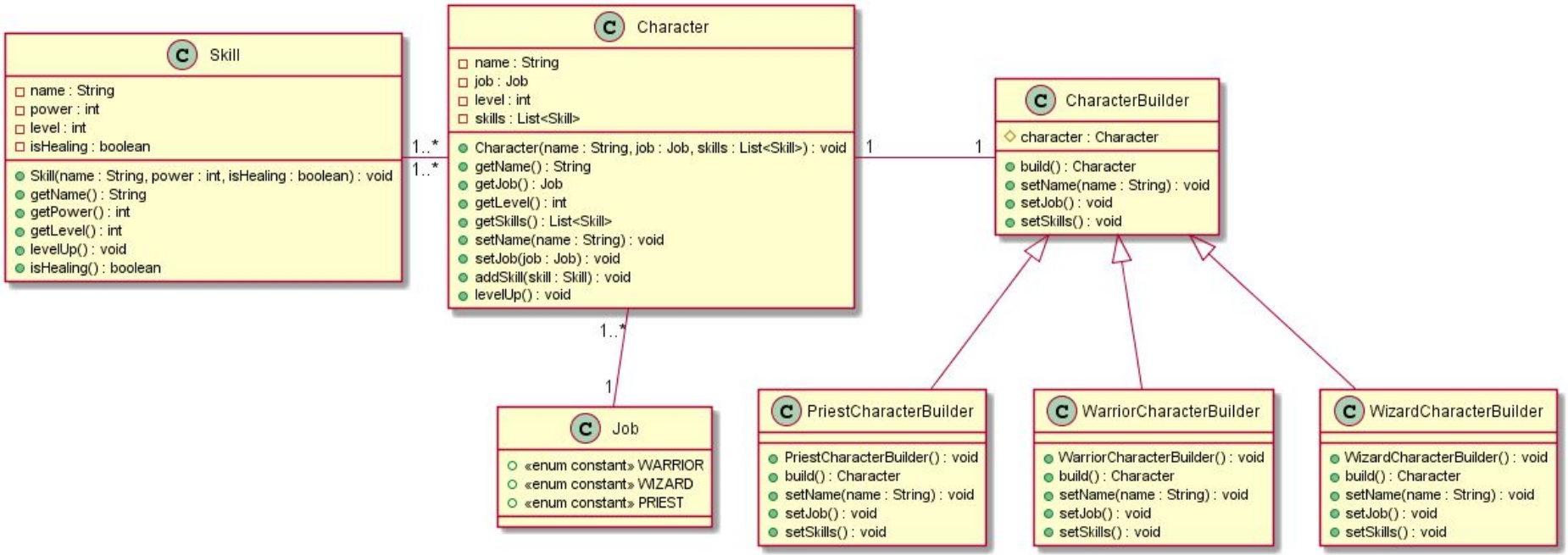
Objetivo do padrão

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



```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 void setName(String name) { this.name = name; }

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

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

    public void levelUp(){ this.level++; }
}
```

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

```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 this.name; }

    public int getPower() { return this.power; }

    public int getLevel(){ return this.level;}

    public boolean isHealing(){ return isHealing; }

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

```java
public abstract class CharacterBuilder {

    protected Character character;

    public abstract Character build();

    public abstract void setName(String name);

    public abstract void setJob();

    public abstract void setSkills() throws Exception;
}
```

```java
public class WizardCharacterBuilder extends CharacterBuilder {
    public WizardCharacterBuilder(){
        this.character = new Character();
    }

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

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

    @Override
    public void setJob() { this.character.setJob(Job.WIZARD); }

    @Override
    public void setSkills() throws Exception {
        character.addSkill(new Skill( name: "Fire ball",  power: 30,  isHealing: false));
        character.addSkill(new Skill( name: "Ice spike",  power: 30,  isHealing: false));
    }
}
```

```java
public class WarriorCharacterBuilder extends CharacterBuilder {

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

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

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

    @Override
    public void setJob() { this.character.setJob(Job.WARRIOR); }

    @Override
    public void setSkills() throws Exception {
        character.addSkill(new Skill( name: "Falcon Punch",  power: 55,  isHealing: false));
    }
}
```

```java
public class PriestCharacterBuilder extends CharacterBuilder {

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

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

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

    @Override
    public void setJob() { this.character.setJob(Job.PRIEST); }

    @Override
    public void setSkills() throws Exception {
        character.addSkill(new Skill( name: "Healing",  power: 10,  isHealing: true));
        character.addSkill(new Skill( name: "Holy light",  power: 20,  isHealing: false));
    }
}
```