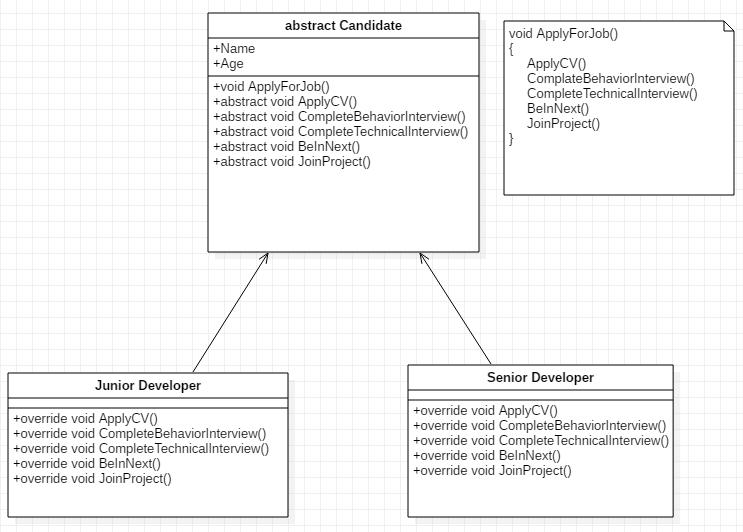
**Template Pattern**

* The skeleton of the algorithm has already introduced and will not change throughout the operation process. The algorithm operates without changing the structures
* Subclasses are able to redefine the certain steps of an algorithm, but changes will not affect the core architecture.
* Q&A questions:
  + In this pattern, subclasses can simply redefine the methodsbased on their needs. Is this understanding correct?
  + In the abstract class Candidate, only some methods areabstract, and the other methods are concrete methods.What is the reason behind this?
  + **Using Hook in implementing Template Pattern**
  + **Key advantages:**
    - **We can control the core flow**
    - **We can define the parts that using for all concreate classes commonly. Avoid duplicate codes.**

****

**Simple Factory Pattern**

1. Why shouldn’t we use *new* operator in FACTORY PATTERN

* No. These are key reasons behind the previous design:
  + One of the key object-oriented design principles is to **separate the parts of your code that are most likely to change from the rest**. In this case, only the creation process for objects changes. You can assume that these animals must speak and perform some actions and that part of code does not need to vary inside the client code. So, in the future, **if there is any change required in the creation process**, **you need to change only the CreateAnimal() method of the SimpleFactory class**. The client code will be unaffected because of those changes.
  + You **do not want to put lots of if-else blocks** (or switch statements) inside the client body. That makes your code clumsy.
  + How you are **creating the objects is hidden from the client code**. This kind of abstraction promotes security

1. What are the challenges associated with this pattern?

* If you want to **add a new animal or delete an existing animal, you need to modify the CreateAnimal() method**. This process will violate the open/closed principle

|  |  |  |
| --- | --- | --- |
| Simple Factory Pattern | Factory Method Pattern | Abstract Factory Pattern |
| * Advantages: * Did not use *if-else* block in client code. * Delegate creation process to an individual class. | * Advantages: * Easy for extension. Just create class extends factory interface. | * Advantages: * Factory of factories |
| * Challenges: * Violate OCP. | * Challenges: * It may affect performance site | * Challenges: * Changes in abstract factory will force to propagate |