**Что такое паттерн Builder и зачем он нужен**

Представим, что нам нужно создать класс с большим количеством параметров - например, класс **Person**. Какие поля у нас будут? Давайте добавим:

* имя (name)
* фамилия (surname)
* возраст (age)
* рост (height)
* вес (weight)
* родители (parents)

Можно еще много параметров дописывать - но и этого пока хватит. В "реальных" классах полей может быть и намного больше.

Таким образом, получаем такой класс:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public class Person {        private String name;      private String surname;      private int age;      private int height;      private int weight;      private Set<Person> parents;    } |

А теперь представим, что мы хотим для данного класса задать **все возможные варианты** конструкторов, которые нам могут понадобиться. Зачем? Чтобы в разных частях кодах использовать **разные** конструкторы и создавать экземпляры класса Person с**разным** количеством заполненных полей ( например, у нас в классе 6 полей. А мы хотим заполнить только 3 поля).

Например, это будет выглядеть так:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | public class Person {        private String name;      private String surname;      private int age;      private int height;      private int weight;      private Set<Person> parents;        Person (String name, String surname){          this.name = name;          this.surname = surname;      }        Person (String name, String surname, int age){          this.name = name;          this.surname = surname;          this.age = age;      }        Person (String name, String surname, int age, int height){          this.name = name;          this.surname = surname;          this.age = age;          this.height = height;      }        //... more constructors here  } |

Как видите, мы прописали 3 конструктора и предполагается, что их может быть намного больше.

И так нам придется писать разные сочетания параметров? Более того, очень легко будет случайно перепутать параметры местами. Например, если мы напишем **Person("Doe", "Jane")** вместо **Person("Jane", "Doe")**- перепутаем местами имя и фамилию? Или, например, возраст и рост? Оба параметра имеют одинаковый тип, и мы можем долго не замечать ошибку.

Все эти проблемы нам помогает решить паттерн проектирования Builder. Давайте создадим его!

Для этого **вернемся к первоначальной версии**нашего класса:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public class Person {        private String name;      private String surname;      private int age;      private int height;      private int weight;      private Set<Person> parents;  } |

Создаем внутри нашего класса еще один класс - **Builder**:

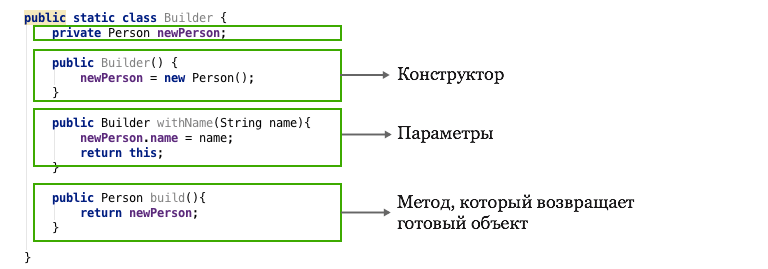
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | public class Person {        private String name;      private String surname;      private int age;      private int height;      private int weight;      private Set<Person> parents;       /\* ---=== getters are supposed to be here. We skipped this part of code to make it simple ===--- \*/        public static class Builder {          private Person newPerson;            public Builder() {              newPerson = new Person();          }            public Builder withName(String name){              newPerson.name = name;              return this;          }            public Builder withSurname(String surname){              newPerson.surname = surname;              return this;          }            public Builder withAge(int age){              newPerson.age = age;              return this;          }            public Builder withHeight(int height){              newPerson.height = height;              return this;          }            public Builder withWeight(int weight){              newPerson.weight = weight;              return this;          }            public Builder withParents(Set<Person> parents){              newPerson.parents = parents;              return this;          }            public Person build(){              return newPerson;          }        }  } |

Многовато? Давайте разберем по частям.

Если бы у нас был такой же **Builder**, только на один параметр (например, **имя**), он выглядел бы так:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public static class Builder {          private Person newPerson;            public Builder() {              newPerson = new Person();          }            public Builder withName(String name){              newPerson.name = name;              return this;          }            public Person build(){              return newPerson;          }      } |

Если разделить на части будет так:



Далее вызываем **Builder** - пишем main():

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public static void main(String[] args) {      Person myPerson = new Person.Builder()              .withName("Jane")              .withSurname("Doe")              .withAge(32)              .withHeight(165)              .withWeight(70)              .build();  } |

Готово! Теперь мы построили наш первый объект с помощью паттерна **Builder**.

Согласитесь - теперь благодаря паттерну Builder:

* код выглядит **гораздо читабельнее**. Ведь все, относящееся к созданию объекта, вынесено в отдельный класс  - Builder;
* при заполнении полей объекта теперь **параметры трудно перепутать**;
* мы можем заполнять**не все параметры** класса. Как Вы могли заметить, мы указали все параметры кроме родителей (parents).