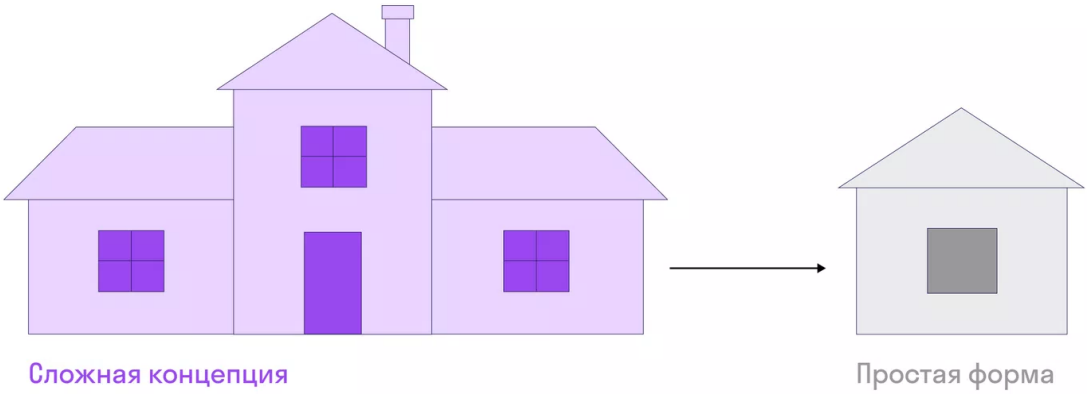
3. Абстрактные классы

Что, если пользователь захочет создать объект `PrintedProduct` ? Ведь у нас не должно быть абстрактных печатных объектов без какого-либо четкого типа (газета, журнал или книга).

Чтобы закрыть дорогу таким ошибкам, нам нужно запретить создавать объект класса `PrintedProduct` и указать, что его можно использовать только в качестве абстрактного носителя общего кода для реализаций.

> Абстракция представляет **сложную концепцию** в более **простой форме**.



В этом нам поможет ключевое слово `abstract` .

Добавим к объявлению класса `PrintedProduct` это ключевое слово. Теперь при попытке создать `new PrintedProduct` Java скажет, что класс является абстрактным и его создать нельзя.

Но на этом особенности абстрактных классов не заканчиваются.

> Если класс **является абстрактным**, это позволяет нам **объявить внутри него абстрактные методы** — методы, которые мы объявляем, но для которых не пишем реализацию.

Это нужно для ситуаций, когда у нас есть общий код, содержащий вызов метода, реализация которого может отличаться в зависимости от наследников.

Например, мы можем написать абстрактный класс, который занимается управлением телевизора, и две его реализации.

Абстрактный класс:

```
public abstract class TVController {

    public void turnOnFirstChannel() {
        turnOnTV();
        // Логика по включению 1-го канала
    }

    public abstract void turnOnTV();

}
```

Две его реализации:

```
public class SmartSpeaker extends TVController {

    @Override
    public void turnOnTV() {
        while (пользователь не сказал слово "включить ТВ") {
            ждать;
        }
    }
}

public class RemoteController extends TVController {

    @Override
    public void turnOnTV() {
        while (пользователь не нажал кнопку включения) {
            ждать;
        }
    }
}
```

Теперь мы можем создать функционал «Работать с обеими реализациями» по ссылке типа TVController. Например:

```
public class Main4 {
   public static void main(String[] args) {

      TVController[] controllers = new TVController[] {
         new SmartSpeaker(),
         new RemoteController()
      };

      for (TVController controller : controllers) {
         controller.turnOnFirstChannel();
      }
   }
}
```

Независимо от способа управления мы включим на всех телевизорах, чьи пульты лежат в массиве, первый канал. Реализация turnOnTV уже будет браться из конкретного объекта с реализацией метода.

Важно запомнить, что **обычный класс не может иметь абстрактных методов**. Это значит, что любой класс, который наследует абстрактный класс с абстрактными методами, обязан их реализовать внутри себя. Или тоже стать абстрактным.