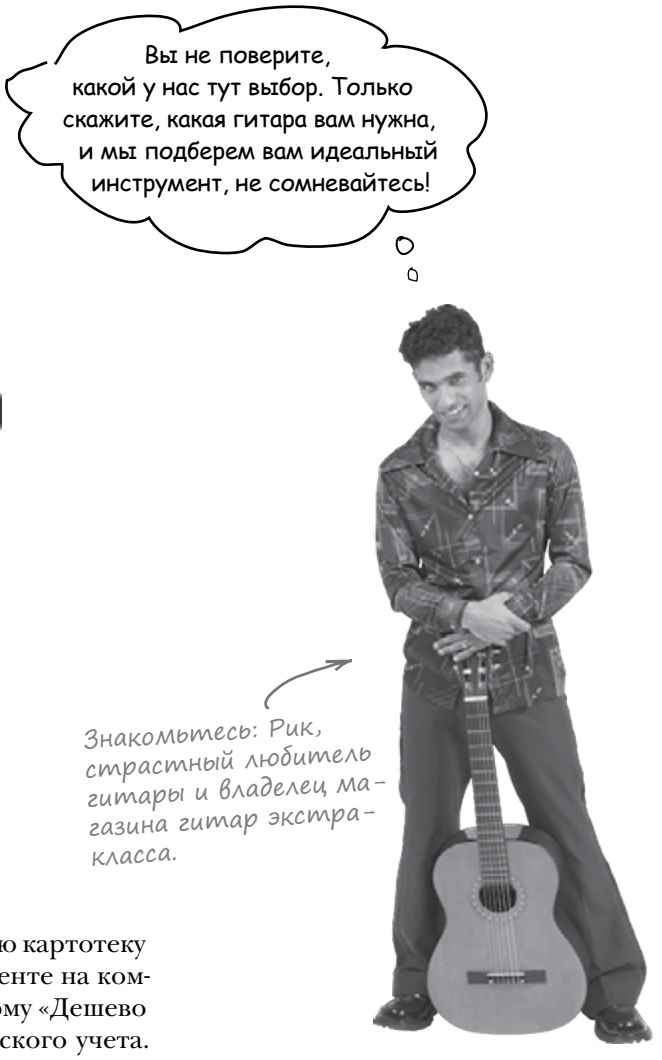
С чего начинаются хорошие программы

Так как же на самом деле пишутся хорошие программы? Всегда сложно понять, с чего следует начинать. Делает ли приложение то, что ему положено делать? И как насчет таких тонкостей, как дублирование кода, — ведь это всегда плохо, верно? Да, определить, **над чем следует работать в первую очередь**, бывает нелегко, к тому же еще нужно не запороть все остальное по ходу дела. Но пусть вас это не тревожит! К тому моменту, когда вы дочитаете эту главу, вы **будете знать**, **как пишутся хорошие программы**, а ваш подход к разработке приложений навсегда изменится. И наконец-то вы поймете, почему ООАП — это такое слово, которое не стыдно произносить в приличном обществе.

Рок-н-ролл навсегда!

Нет ничего лучше звука классной гитары в руках великого музыканта. Фирма Рика специализируется на поиске идеальных инструментов для своих требовательных покупателей.

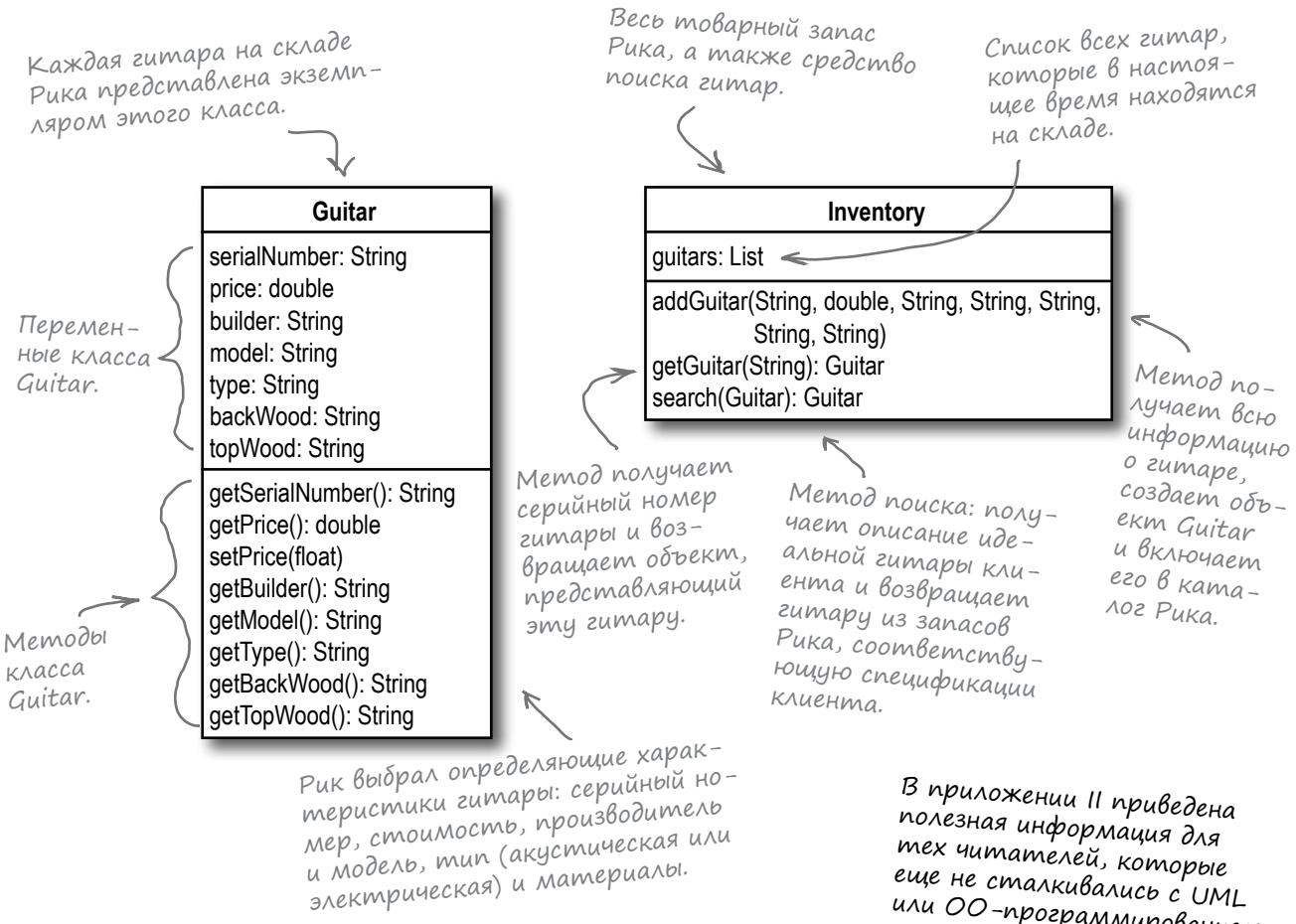


Знакомьтесь: Рик, страстный любитель гитары и владелец магазина гитар экстра-класса.

Несколько месяцев назад Рик решил выбросить свою картотеку и начал хранить информацию о текущем ассортименте на компьютере. Он обратился с заказом в популярную фирму «Дешево и сердито», и там ему уже написали систему складского учета. Рик даже вытребовал средство поиска, которое поможет ему подобрать для каждого клиента «инструмент мечты».

Новенькое приложение Рика...

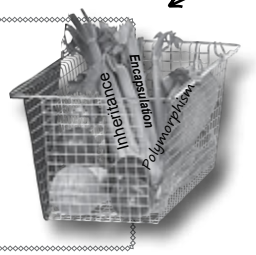
А вот и приложение, которое Рик получил от разработчиков из фирмы «Дешево и сердито». Построенная ими система полностью заменяет все рукописные заметки Рика и помогает ему подобрать идеальную гитару для каждого клиента. Чтобы показать, что было сделано, разработчики вручили Рику следующую диаграмму классов UML:



В приложении II приведена полезная информация для тех читателей, которые еще не сталкивались с UML или ОО-программированием.

Недавно в Объективиле?

Если вы не разбираетесь в объектно-ориентированном программировании, никогда не слышали о UML или не уверены в том, что означает приведенная диаграмма, — ничего страшного! Мы подготовили специальный краткий курс для новичков. Загляните в конец книги и прочитайте приложение II — обещаем, вы не пожалеете. А потом возвращайтесь сюда, и все происходящее покажется вам куда более осмысленным.



А вот как выглядит код Guitar.java

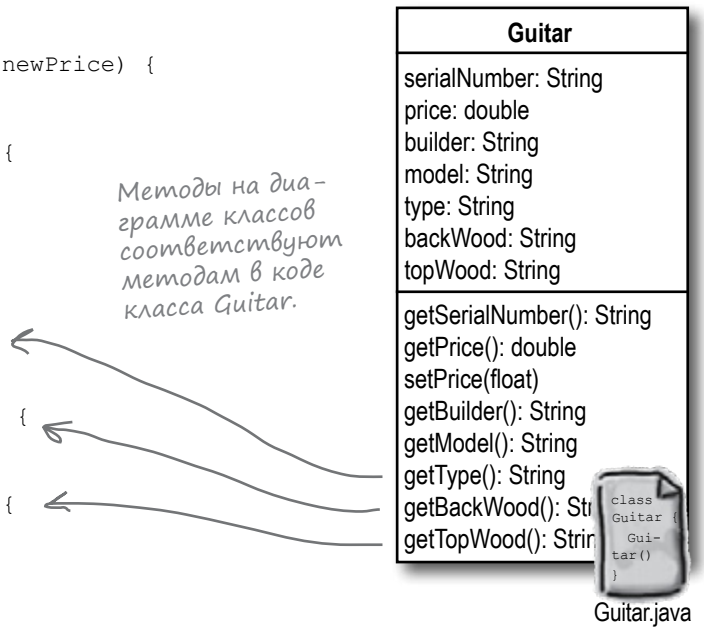
Диаграмма классов приложения Рика была приведена на предыдущей странице; а теперь давайте посмотрим, как выглядит реальный код **`Guitar.java`** и **`Inventory.java`**.

Свойства, которые мы видели на диаграмме класса Guitar.

```
public class Guitar {

  private String serialNumber, builder, model, type, backWood, topWood;
  private double price;

  public Guitar(String serialNumber, double price,
                String builder, String model, String type,
                String backWood, String topWood) {
    this.serialNumber = serialNumber;
    this.price = price;
    this.builder = builder;
    this.model = model;
    this.type = type;
    this.backWood = backWood;
    this.topWood = topWood;
  }

  public String getSerialNumber() {
    return serialNumber;
  }

  public double getPrice() {
    return price;
  }
  public void setPrice(float newPrice) {
    this.price = newPrice;
  }
  public String getBuilder() {
    return builder;
  }
  public String getModel() {
    return model;
  }
  public String getType() {
    return type;
  }
  public String getBackWood() {
    return backWood;
  }
  public String getTopWood() {
    return topWood;
  }
}
```

На диаграммах классов UML конструкторы не отображаются. Впрочем, конструктор Guitar делает ровно то, что можно от него ожидать: он задает все свойства нового объекта Guitar.

Методы на диаграмме классов соответствуют методам в коде класса Guitar.



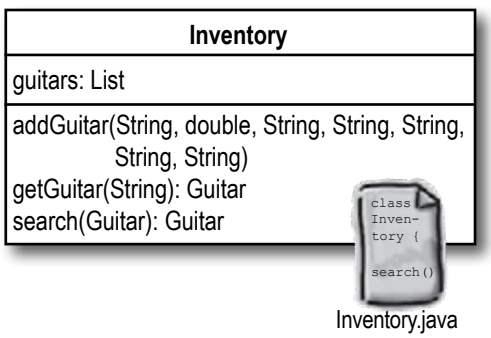
И Inventory.java...

```
public class Inventory {
  private List guitars;

  public Inventory() {
    guitars = new LinkedList();
  }

  public void addGuitar(String serialNumber, double price,
                        String builder, String model,
                        String type, String backWood, String topWood) {
    Guitar guitar = new Guitar(serialNumber, price, builder,
                               model, type, backWood, topWood);
    guitars.add(guitar);
  }
  public Guitar getGuitar(String serialNumber) {
    for (Iterator i = guitars.iterator(); i.hasNext(); ) {
      Guitar guitar = (Guitar)i.next();
      if (guitar.getSerialNumber().equals(serialNumber)) {
        return guitar;
      }
    }
    return null;
  }
  public Guitar search(Guitar searchGuitar) {
    for (Iterator i = guitars.iterator(); i.hasNext(); ) {
      Guitar guitar = (Guitar)i.next();
      // Серийный номер игнорируется, так как он уникален
      // Цена игнорируется, так как она уникальна
      String builder = searchGuitar.getBuilder();
      if ((builder != null) && (!builder.equals("")) &&
          (!builder.equals(guitar.getBuilder())))
        continue;
      String model = searchGuitar.getModel();
      if ((model != null) && (!model.equals("")) &&
          (!model.equals(guitar.getModel())))
        continue;
      String type = searchGuitar.getType();
      if ((type != null) && (!searchGuitar.equals("")) &&
          (!type.equals(guitar.getType())))
        continue;
      String backWood = searchGuitar.getBackWood();
      if ((backWood != null) && (!backWood.equals("")) &&
          (!backWood.equals(guitar.getBackWood())))
        continue;
      String topWood = searchGuitar.getTopWood();
      if ((topWood != null) && (!topWood.equals("")) &&
          (!topWood.equals(guitar.getTopWood())))
        continue;
    }
    return null;
  }
}
```

Не забудьте, что директивы импорта были опущены для экономии места.

addGuitar() получает все свойства, необходимые для создания нового экземпляра Guitar, создает его и включает в каталог.

Метод работает довольно примитивно... Каждое свойство полученного объекта Guitar сравнивается с аналогичным свойством каждого объекта Guitar в каталоге Рика.



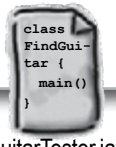
А потом от Рика стали уходить клиенты...

Похоже, независимо от запросов клиента новая поисковая программа Рика всегда выдает пустой результат. Но Рик совершенно точно знает, что у него на складе есть подходящая гитара... Что происходит?

FindGuitarTester.java имитирует типичный рабочий день Рика. Приходит клиент, говорит, что ему нужно, — Рик проводит поиск по каталогу.

```
public class FindGuitarTester {

  public static void main(String[] args) {
    // Инициализация каталога гитар
    Inventory inventory = new Inventory();
    initializeInventory(inventory);

    Guitar whatErinLikes = new Guitar("", 0, "fender", "Stratocastor",
                                      "electric", "Alder", "Alder");
    Guitar guitar = inventory.search(whatErinLikes);
    if (guitar != null) {
      System.out.println("Erin, you might like this " +
        guitar.getBuilder() + " " + guitar.getModel() + " "
        guitar.getType() + " guitar:\n   " +
        guitar.getBackWood() + " back and sides,\n   " +
        guitar.getTopWood() + " top.\nYou can have it for only $" +
        guitar.getPrice() + "!");
    } else {
      System.out.println("Sorry, Erin, we have nothing for you.");
    }
  }

  private static void initializeInventory(Inventory inventory) {
    // Включение описаний гитар в каталог.
  }
}
```

Эрин ищет гитару фендер «Стратокастер», сделанную из ольхи (Alder).



FindGuitarTester.java

```
File Edit Window Help C7#5
%java FindGuitarTester
Sorry, Erin, we have nothing for you.
```

Вот что происходит, когда Эрин приходит в магазин, а Рик пытается подобрать ей гитару.

Но я точно знаю, что у меня есть такая гитара! Смотрите, да вот же она:

```
inventory.addGuitar("V95693",
  1499.95, "Fender", "Stratocastor",
  "electric", "Alder", "Alder");
```

И все характеристики совпадают с тем, что хочет Эрин... В чем дело?

Фрагмент кода инициализации каталога Рика. Действительно, у него есть идеальная гитара для Эрин.

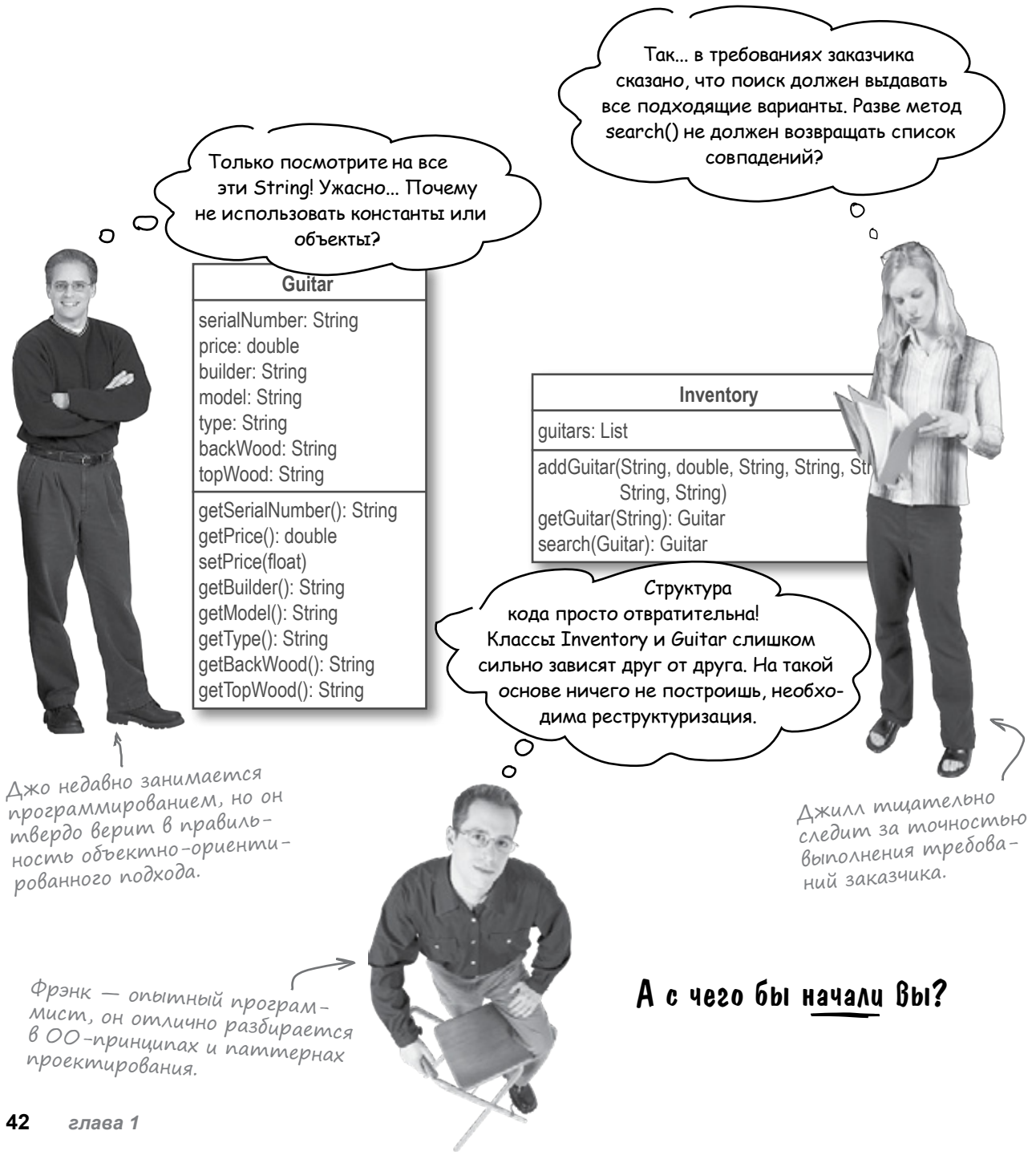
Возьми в руку карандаш

Как бы вы переработали приложение Рика?

Просмотрите код приложения на трех последних страницах и результаты поиска. Какие недостатки вы заметили? Что бы вы изменили в этом коде? Запишите внизу ПЕРВОЕ, что бы вы сделали для усовершенствования приложения Рика.

Что бы вы изменили В ПЕРВУЮ ОЧЕРЕДЬ?

Очевидно, в приложении Рика есть проблемы — но пока не совсем понятно, с чего начинать. А выбор широк, прямо глаза разбегаются:



А с чего бы начали Вы?

Откуда я знаю, с чего начинать? В каждом новом проекте, над которым я работаю, у всех разные мнения по поводу того, что делать в первую очередь. Иногда я угадываю верно, иногда приходится переделывать все приложение из-за того, что я начал не с того. А я просто хочу писать хорошие программы! Так с чего нужно начать в приложении Рика?

Как писать хорошие программы каждый раз?

Одну минуту... Извините, что встреваю в разговор, но что такое «хорошие программы»? Звучит слишком неопределенно, вы не находите?

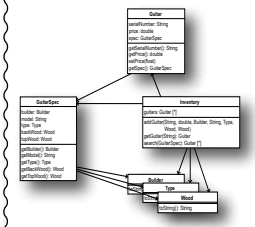
Хороший вопрос... и на него есть много разных ответов:

Программист, внимательно относящийся к заказчику:

«Хорошая программа всегда делает то, что хочет заказчик. Даже если заказчик захочет использовать программу каким-то новым способом, программа не должна сбоить или выдавать неожиданные результаты».

При таком подходе главное — чтобы заказчик был доволен тем, как работает приложение.

Объектно-ориентированный программист:



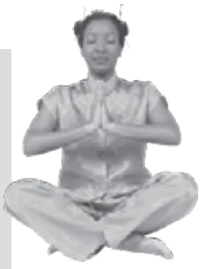
«Хорошая программа — это объектно-ориентированный код. Он не содержит дубликатов, а каждый объект более или менее полно управляет своим поведением. Такие программы легко расширять, потому что они имеют надежную и гибкую структуру кода».

При структурно-ориентированном подходе код оптимизируется для расширения и повторного использования, в нем используются паттерны проектирования и проверенные ОО-методологии.

Хороший ОО-программист всегда старается сделать свой код более гибким.

Гуру в области проектирования:

«Хорошая программа строится с использованием проверенных паттернов и принципов проектирования. Объекты обладают слабой связностью; код открыт для расширения, но закрыт для изменения. Такая структура кода также упрощает его повторное использование, поскольку вам не приходится переписывать все заново, чтобы использовать готовые части своего приложения».



Пытаетесь понять, что все это значит? Ничего страшного... узнаете в следующих главах.