Абстрактные классы и методы

- Абстрактный класс не предполагает создание экземпляров.
- Абстрактный класс может содержать (и не содержать) абстрактные методы и свойства.
- Создавать объекты абстрактных типов **запрещено.**

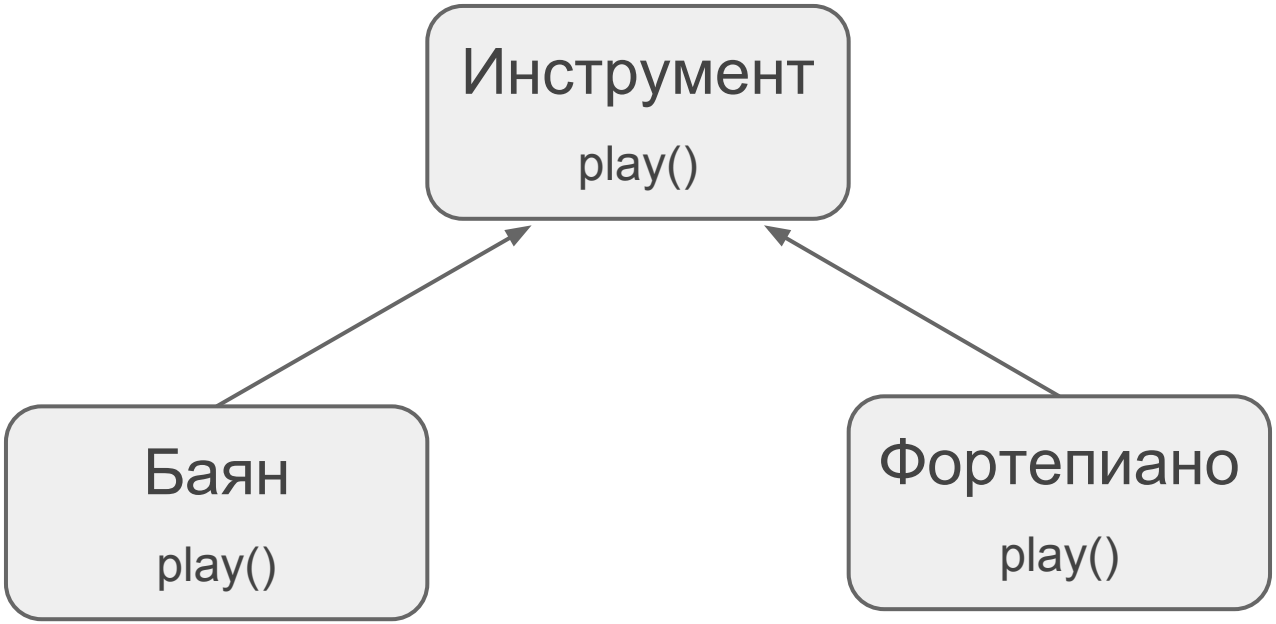
- Класс с абстрактными методами **обязан** быть абстрактным.
- Дочерний класс может перекрыть родительские реализованные методы своими абстрактными.
- Абстрактный класс не обязан иметь абстрактные методы.

Абстрактные классы и методы

- Абстрактные методы описывают сигнатуру без реализации:

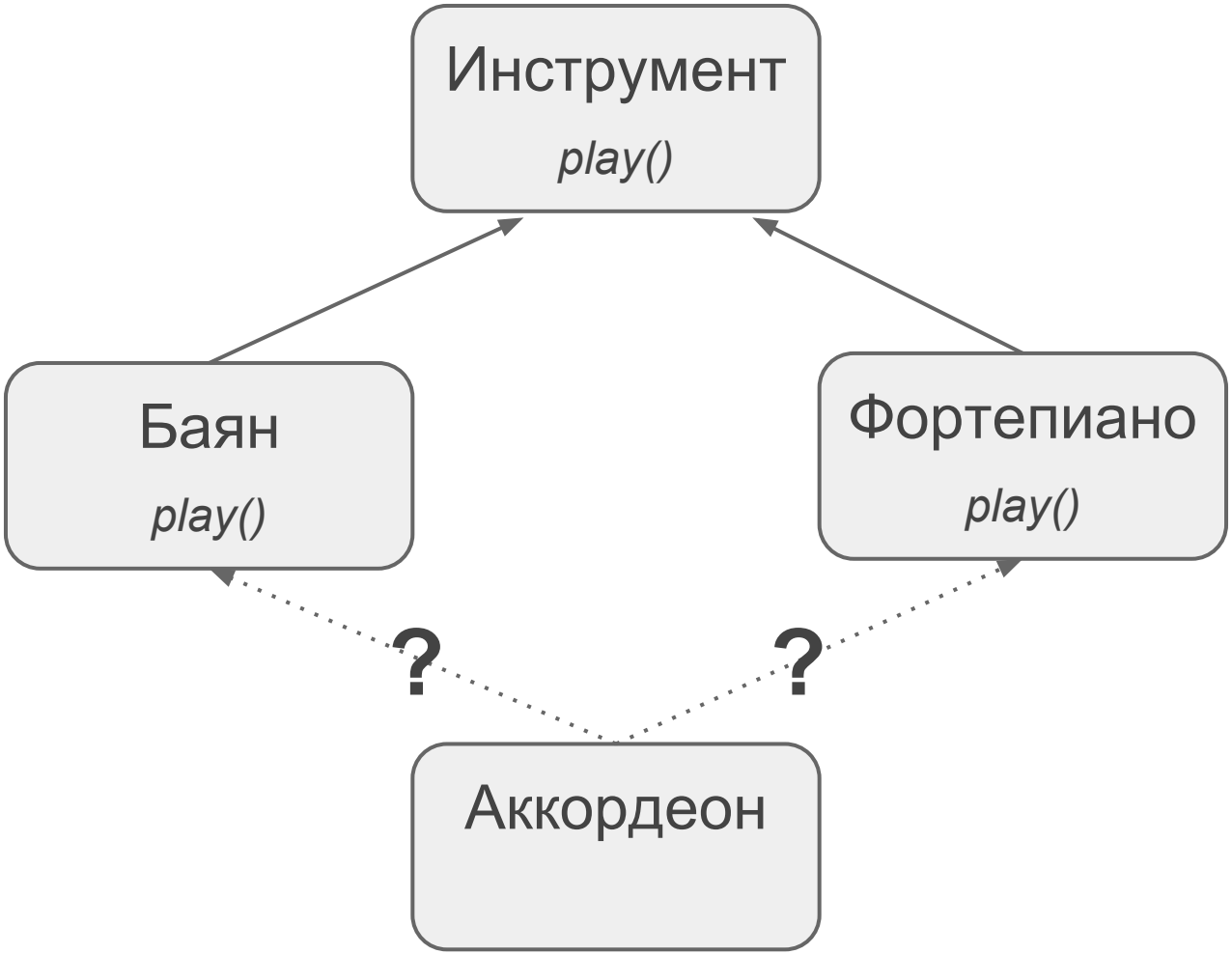
```
abstract public int getValue();
```

Проблема множественного наследования



Проблема множественного наследования

Ромбовидное наследование



Интерфейсы

- Используются для специфицирования услуг, предоставляемых классом: показывают, **что** должны делать классы, не описывая, **как** это делать.

- Позволяют описать тип в полностью абстрактной форме.

- Не являются классами. Однако, с точки зрения реализации, интерфейс - это чистый абстрактный класс, в котором не определено ничего, кроме абстрактных методов.

- Классы способны *реализовывать* один или несколько интерфейсов.

Контракт класса

- Набор методов и полей класса, открытых для доступа извне тем или иным способом, в совокупности с описанием их назначения.

- Способ выражения обещаний автора относительно того, на что способен и для чего предназначен созданный им продукт.

Наследование в Java

Виды наследования:

- Класс:
  - расширяет класс;
  - реализует интерфейсы.

Наследование как внешнего контракта, так и реализации

- Интерфейс:
  - расширяет интерфейсы.

Наследование **только** внешнего контракта

Объявление интерфейсов

```
interface Somethingable {
    // константы
    // методы
    // вложенные классы и интерфейсы
}
```

- Все члены интерфейса по умолчанию обладают признаком **public**. Применение других модификаторов не имеет смысла.
- Поля, объявленные в интерфейсе имеют модификаторы **public static.**
- Встречаются пустые интерфейсы.

Константы в интерфейсах

```
interface Verbose {
    int SILENT = 0;
    int TERSE = 1;
    int NORMAL = 2;
}
```

- Имеют неявные модификаторы **public static final**
- Должны быть инициализированы.

Методы в интерфейсах

```
interface Verbose {
    void setVerbosity(int level);
    int getVerbosity();
}
```

- Имеют неявные модификаторы **public abstract**
- Не могут иметь модификаторов:
  - **native synchronized**
  - **strictfp static final**

Расширение интерфейсов интерфейсами

```
interface NewVerbose extends Verbose, Runnable {
    // ...
}
```

- Допускается сокрытие констант.
- Переопределение метода не несет семантической нагрузки.
- Совпадение имен наследуемых методов не несет семантической нагрузки.

Реализация интерфейсов классами

```
class DataThread extends MyThread
        implements Runnable, Verbose
{
        // ...
}
```

- Интерфейсы реализуются классами.
- Класс может реализовывать несколько интерфейсов.
- Если класс не реализует все методы «наследуемых» интерфейсов, он является абстрактным.

Интерфейс или абстрактный класс?

- Интерфейсы обеспечивают инструментарий множественного наследования, производный класс способен наследовать одновременно **несколько** интерфейсов.

- Класс может расширять **единственный** базовый класс, даже если тот содержит только абстрактные методы.

Интерфейс или абстрактный класс?

- Абстрактный класс частично может быть реализован, он вправе содержать члены, помеченные как **protected** и/или **static** и т.п.

- Структура интерфейса ограничена объявлениями **public**-констант и **public**-методов без какой бы то ни было реализации.

- Основное назначение интерфейсов - возможность простой замены реализации, а не введение лишнего уровня абстракции.

Ссылки интерфейсных типов

- Допускаются ссылки интерфейсных типов.

- Такая ссылка позволяет выполнять над объектом операции, описанные во внешнем контракте, обусловленном типом интерфейса.

- Такое средство существенно расширяет возможности полиморфизма.

Пустые интерфейсы

- Существуют пустые интерфейсы (интерфейсы-маркеры), объявления которых не содержат ни констант, ни методов.

- Реализация маркерных интерфейсов обычно означает способность объекта к чему-либо.

- Ссылка такого типа редко имеет смысл, поскольку внешний контракт пуст.

- Даже такая ссылка позволяет выполнять методы объекта, а именно методы, объявленные в классе *Object*, поскольку они есть у абсолютно любого объекта.