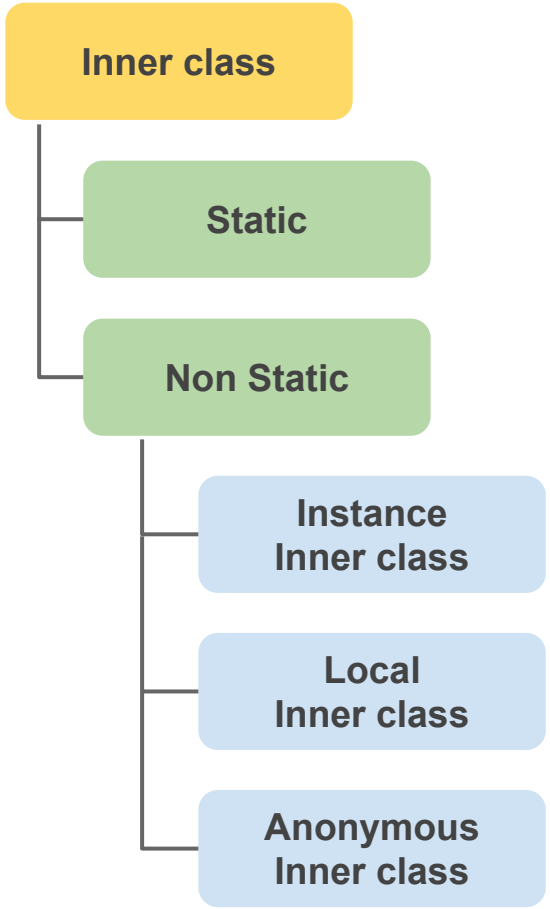
Внутренние классы

- Определение класса находится внутри другого.
- Группировка классов, логически принадлежащих друг другу.
- Не является композицией.

- Делятся на:
  - *вложенные* (“nested”)
  - *внутренние* (“inner”)

- Являются членами внешнего класса и могут иметь различные уровни доступа.

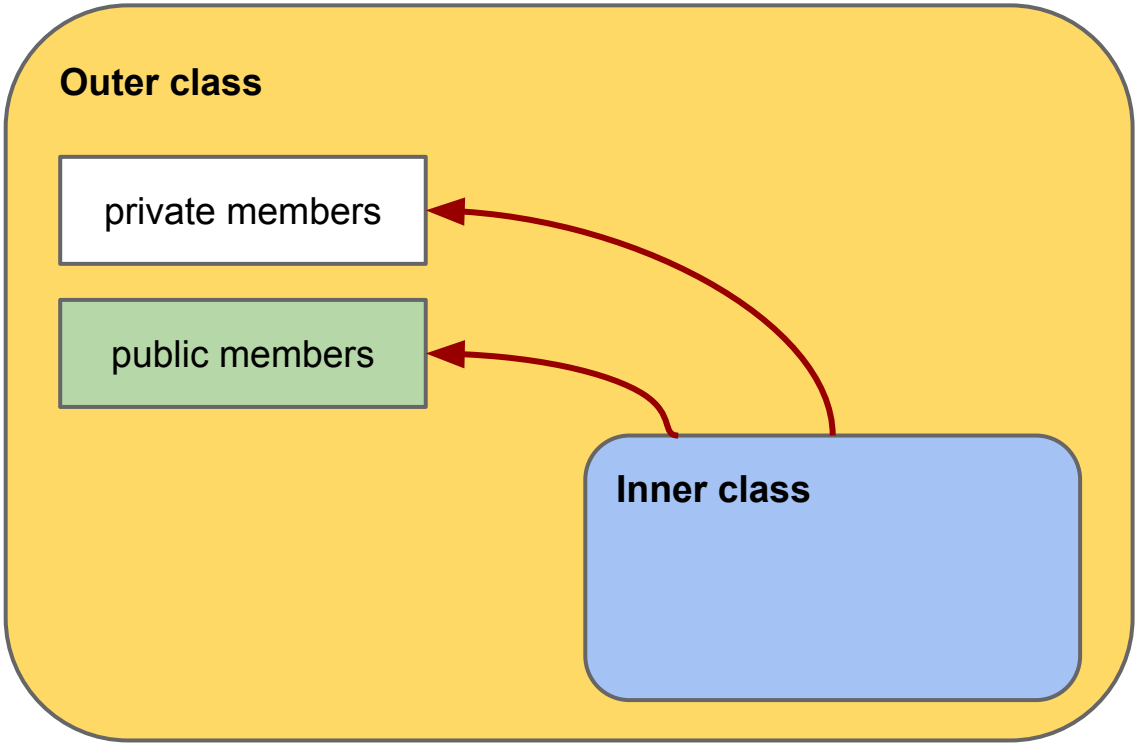
Типы внутренних классов



В Java возможно создание внутренних классов нескольких типов:

- Статические вложенные классы.
- Нестатические внутренние классы.
- Локальные классы.
- Анонимные классы.

Доступ к приватным членам класса



Внутренний класс может обращаться ко всем полям и методам внешнего класса, как будто они описаны в нем самом.

Содержит **скрытую ссылку на объект** внешнего класса:

Outer.this

Статический вложенный класс

```
public class Outer {
    public static class Nested {
    }
}
```

Внутренний класс можно сделать статическим, если не нужна связь между объектами классов.

+ Для создания объекта статического вложенного класса не нужен объект внешнего класса
+ Может содержать статические поля и методы
- Из объекта вложенного класса нельзя обращаться к не-статическим членам внешнего класса
- Нет ссылки на объект внешнего класса

```
// Создание объекта вложенного класса
Outer.Nested  instance = new Outer.Nested();
```

Внутренний класс

```
public class Outer {
    public class Inner {
    }
}
```

Внутренний класс **всегда** ассоциирован с объектом внешнего класса.

+ Внутренний класс имеет доступ к полям и методам внешнего класса, даже если они закрытые (private)
- Не может содержать статические поля и методы

Сокрытие полей и методов

В случае, когда во внутреннем классе определено поле или метод с тем же именем, что и во внешнем, то из внутреннего класса можно получить доступ только к собственным одноименным полям или методам.

Доступ к одноименным полям или методам внешнего класса можно получить с помощью ссылки на объект этого класса:

Имя класса + .this. + *имя поля*

```
Outer.this.field;
Outer.this.method();
```

Локальный класс

```
public class Outer {
    public void printText() {
        class Local {
        }

        Local local = new Local();
    }
}
```

Разновидность внутреннего класса (не-статического), определение которого находится внутри метода или блока { … }

- Доступен только внутри метода или блока, в котором задан
- Имеет доступ к локальным неизменяемым (*final* или “фактически” final) переменным внутри блока
- Методы и поля не могут быть статическими, за исключением констант (*static final*)

Анонимный класс

Представляет собой внутренний класс, не имеющий имени. Может быть объявлен как подкласс существующего класса или как реализация некоего интерфейса.

Описание анонимного класса связано с его инстанцированием.

```
public class SuperClass{
    public void printText() { … }
}

SuperClass instance = new SuperClass() {
    public void printText() { … }
};

instance.printText();
```

Реализация интерфейса классом

Анонимный класс способен реализовывать интерфейс, вместо расширения другого класса.

Имеет доступ к членам внешнего класса и к неизменяемым локальным переменным.

```
public interface MyInterface{
    void printText() { … }
}

MyInterface instance = new MyInterface() {
    public void printText() { … }
};

instance.printText();
```

Как насчет конструктора?

Внутри анонимного класса можно объявлять поля и методы, но **нельзя создавать конструкторы.**

Однако возможно объявление блока инициализации.

```
final Strint PLANET = "Earth";

MyInterface instance = new MyInterface() {
    private String text;
    {  this.text = PLANET;  }

    public void printText() {
        System.out.println(this.text);
    }
}
```

Идентификаторы классов

Каждый класс компилируется в файл с расширением *.class*, содержащий полную информацию о создании его экземпляров.

Имена внутренних классов строятся по следующей схеме:

OuterClassName$InnerClassName

Для анонимных внутренних классов компилятор использует номера в качестве их идентификаторов.

OuterClassName$1
OuterClassName$1Local