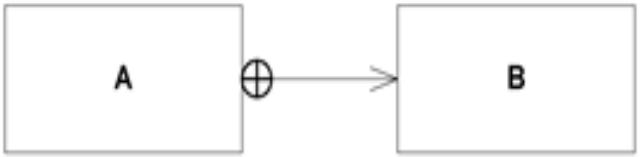
Модуль #7
Расширенные вопросы

Модуль 7 Расширенные вопросы

- **Вложенные классы**
- Статические вложенные классы
- Классы, определенные внутри метода
- Анонимные классы
- Конвертация объектных ссылок при присваивании, вызове метода
- Конвертация и приведение объектных ссылок
- Assertions

Вложенные классы

- Вложенный класс – это класс, объявленный в рамках описания другого класса.
- Класс можно объявить в любом блоке, включая блоки, являющиеся частью метода.
- Вложенные классы отличаются своей видимостью и областью доступа.




```
public class A {
  private class B {
    ...
  }
}
```

Внимание! Классы, вложенные в метод, называются локальными.

Пример вложенного класса

```
public class OuterOne {
    private int x;
    public class InnerOne {
        private int y;
        public void innerMethod() {
            System.out.println("enclosing x is " + x);
            System.out.println("y is " + y);
        }
    }
    public void outerMethod() {
        System.out.println("x is " + x);
    }
    public void makeInner() {
        InnerOne anInner = new InnerOne();
        anInner.innerMethod();
    }
    // other methods...
}
```

Вложенные классы

- Полное имя вложенного класса в примере `OuterOne.InnerOne`.
- После компиляции создается отдельный файл с именем `OuterOne$InnerOne.class`.
- Для создания внутреннего класса, необходим экземпляр внешнего класса, который выступает в роли контекста.

Внимание! Все атрибуты внешнего класса доступны вложенному классу вследствие существования ссылки вложенного класса на внешний (позже в этом модуле).

Вложенные классы

- Создать экземпляр вложенного класса можно только через экземпляр внешнего класса.

```
public static void main(String[] args)
{
  OuterOne.InnerOne i = new OuterOne().new InnerOne();
  i.innerMethod();

  OuterOne outer = new OuterOne();
  OuterOne.InnerOne inner = outer.new InnerOne();
}
```

- Вложенный класс может быть `public`, `private` или `protected`.

Модуль 7 Расширенные вопросы

- Вложенные классы
- **Статические вложенные классы**
- Классы, определенные внутри метода
- Анонимные классы
- Конвертация объектных ссылок при присваивании, вызове метода
- Конвертация и привидение объектных ссылок
- Assertions

Статические вложенные классы

- Вложенный класс может быть помечен как **`static`**.
- Вложенный статический класс не может использовать **`this`** для доступа к атрибутам внешнего объекта.
- Может обращаться к статическим переменным и методам внешнего класса.

```
public class MyOuter {
    public static class MyInner {
    }

    public static void main(String[] args) {
        MyInner aMyInner = new MyOuter.MyInner();
    }
}
```

Модуль 7 Расширенные вопросы

- Вложенные классы
- Статические вложенные классы
- **Классы, определенные внутри метода**
- Анонимные классы
- Конвертация объектных ссылок при присваивании, вызове метода
- Конвертация и привидение объектных ссылок
- Assertions

Классы внутри метода

- Все что объявлено внутри метода – не является членом класса.
- Локальные объекты не могут иметь модификаторов доступа и быть объявленными как `static`.
- Возможно создание анонимных классов.
- Анонимный класс может иметь доступ к внешним объектам, только если они объявленны как `final`.

Классы внутри метода

```
public class MOuter {
    private int m = (int) (Math.random() * 100);
    public static void main(String args[]) {
        MOuter that = new MOuter();
        that.go((int) (Math.random() * 100), (int) (Math.random() *
100));
    }
    public void go(int x, final int y) {
        int a = x + y;
        final int b = x - y;
        class MInner {
            public void method() {
                System.out.println("m is " + m);
                // System.out.println("x is " + x); //Illegal!
                System.out.println("y is " + y);
                // System.out.println("a is " + a); //Illegal!
                System.out.println("b is " + b);
            }
        }
        MInner that = new MInner();
        that.method();
    }
}
```

Модуль 7 Расширенные вопросы

- Вложенные классы
- Статические вложенные классы
- Классы, определенные внутри метода
- **Анонимные классы**
- Конвертация объектных ссылок при присваивании, вызове метода
- Конвертация и привидение объектных ссылок
- Assertions

Анонимные классы

- Иногда класс, определенный внутри метода, не нуждается в имени.

- Анонимный класс может быть объявлен как расширяющий другой, или как реализующий интерфейс.

Пример:

- Закрытие пользовательского окна по клику на кнопке “Close”

- Реакция оформлена в виде отдельного класса:

```
class CloseActionListener implements ActionListener
```

Анонимные классы

```
public class Xbis {
    public static void main( String[] args){
        JFrame frame = new JFrame("Closing example");
        JButton btnClose = new JButton("Close");

        CloseActionListener al = new CloseActionListener();
        btnClose.addActionListener( al ); // .addActionListener()

        frame.add(btnClose);
        frame.pack();
        frame.setVisible(true);
    } // main
} // class X

class CloseActionListener implements ActionListener{
    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("Closing...");
        System.exit(0);
    }
}
```

Анонимные классы

```
public static void main( String[] args){
    JFrame frame = new JFrame("Closing example");
    JButton btnClose = new JButton("Close");
    btnClose.addActionListener(
            new ActionListener(){

            @Override
            public void actionPerformed(ActionEvent e){
                    System.out.println("Closing...");
                    System.exit(0);
                }
            } // new ActionListener
            ); // .addActionListener()

    frame.add(btnClose);
    frame.pack();
    frame.setVisible(true);
} // main
```

Внимание! Нельзя использовать наследование и имплементацию одноверменно.

Анонимные классы

- Анонимные классы удобны, чтоб избежать придумывания тривиальных имен для классов.
- Код класса содержит несколько строк.
- При компиляции создается класс, который называется `EnclosingClassName$n`, где `n` — порядковый номер анонимного класса.

Анонимные классы

```
public void aMethod() {
    theButton.addActionListener(
      new ActionListener() {
          public void actionPerformed(ActionEvent e) {
           System.out.println("The action has
occurred");
          }
      }
   );
}
```

- Для анонимных классов нельзя определить конструктор.
- Можно вызвать конструктор родительского класса.

Анонимные классы

```
public class X extends Frame {
 private int count;
 public X() {
   final Label l = new Label("Count = " + count);
   add(l, BorderLayout.SOUTH);
   add(
     new Button("Hello " + 1) {
        // initializer
         addActionListener(
           new ActionListener() {
             public void actionPerformed(
              ActionEvent ev) {
              count++;
              l.setText("Count = " + count);
            }
           }
        );
     }
   }, BorderLayout.NORTH
);   } }
```

Упражнение 10

Работа с вложенными классами.

Модуль 7 Расширенные вопросы

- Вложенные классы
- Статические вложенные классы
- Классы, определенные внутри метода
- Анонимные классы
- **Конвертация объектных ссылок при присваивании, вызове метода**
- Конвертация и приведение объектных ссылок
- Assertions

Конвертация объектных ссылок

- Конвертация объектной ссылки возникает, когда значение присваивается переменной другого типа.

- 3 вида объектной ссылки:
  - Класс (`Button`, `FileWriter`)
  - Интерфейс (`java.util.List`)
  - Массив (`int[][]`, `TextArea[]`)

```
  Oldtype x = new Oldtype();
Newtype y = x; //конвертация
```

Внимание! Компиляция производится автоматически, когда операция присваивания ”раширяющая”

Конвертация объектных ссылок

Конвертация Oldtype к Newtype

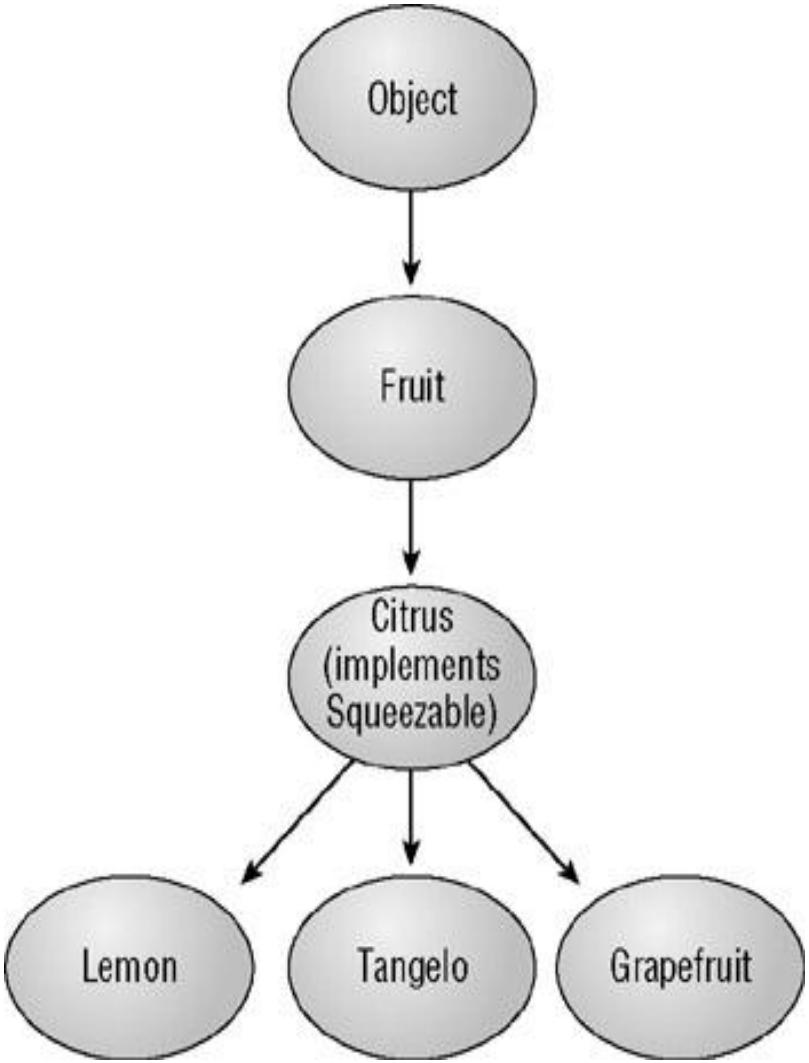
	Oldtype - класс	**Oldtype - интерфейс**	**Oldtype - массив**
Newtype - класс	**`Oldtype`** должен быть подклассом **`Newtype`**	**`Newtype`** должен быть **`Object`**	**`Newtype`** должен быть **`Object`**
Newtype - интерфейс	**`Oldtype`** должен имплементировать интерфейс **`Newtype`**	**`Oldtype`** должен быть подинтерфейсом интерфейса **`Newtype`**	**`Newtype`** должен быть **`Cloneable`** или **`Serializable`**
Newtype - массив	Ошибка компилятора	Ошибка компилятора	**`Oldtype`** должен быть массивом ссылок на объекты, которые могут быть сконвертированы к типу составляющему массив **`Newtype`**

Внимание! Тип `NewType` называется совместимым, при выполнении указанных выше условий.

Правила конвертации

- Интерфейс может быть сконвертирован только в:
  - Родительский (`superinterface`)
  - `Object`
- Класс может быть конвертирован в класс или интерфейс. При конвертации новый тип должен быть базовым по отношению к старому.
- Массив может быть конвертирован к `Object`, `Cloneable`, `Serializable` или массиву(только массив объектных ссылок).

Правила конвертации



Правила конвертации

```
Tangelo tange = new Tangelo();
Citrus cit = tange; // Citrus - класс и
наследует Citrus
--------------------
Citrus cit = new Citrus();
Tangelo tange = cit; //Tangelo - класс и не
наследует Citrus
--------------------
Grapefruit g = new Grapefruit();
Squeezable squee = g; // No problem
Grapefruit g2 = squee; // Error
--------------------
Fruit fruits[];
Lemon lemons[];
Citrus citruses[] = new Citrus[10];
for (int i = 0; i < 10; i++) {
    citruses[i] = new Citrus();
}
fruits = citruses;    // No problem
lemons = citruses;    // Error
```

Правила конвертации

- Те же правила конвертации работают и при вызове метода.

Пример:

- т.к. метод `add()` класса `ArrayList` принимает `Object`, то передать можно:
  - Любой класс
  - Любой интерфейс
  - Любой массив

```
ArrayList myList = new ArrayList();
Tangelo tange = new Tangelo();
myList.add(tange);
```

Модуль 7 Расширенные вопросы

- Вложенные классы
- Статические вложенные классы
- Классы, определенные внутри метода
- Анонимные классы
- Конвертация объектных ссылок при присваивании, вызове метода
- **Конвертация и приведение объектных ссылок**
- Assertions

Конвертация объектных ссылок

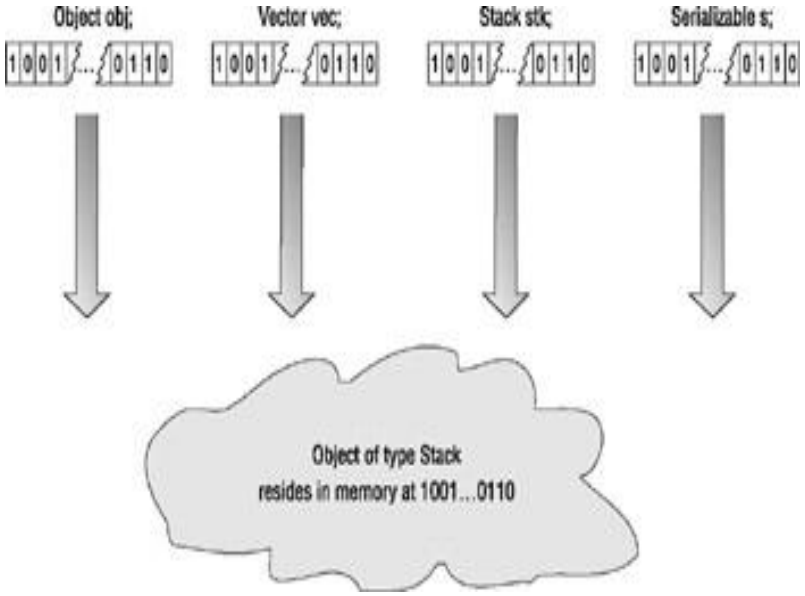
- Приведение (кастинг, cast) объектной ссылки похоже на приведение примитивов: «Программист убеждает компилятор выполнить преобразование, которое может быть запрещено».

- Все виды разрешенной конвертации можно приводить явно.

```
Lemon lem = new Lemon();
Citrus cit = (Citrus)lem;
```

Конвертация объектных ссылок

- Работа с объектами в Java ведется через ссылки, но на объект можно ссылаться с помощью ссылок разного типа.



- Тип ссылочной переменной известен при компиляции, однако класс объекта может быть неизвестен до `runtime`.

Конвертация объектных ссылок

- При выполнении cast компилятор следит, чтобы:
  - `OldType` и `NewType` – были в одной иерархии наследования.
  - Если `OldType` и `NewType` – массивы, то их элементы должны удовлетворять данным условиям.

Внимание! JVM проверит, если полученный объект не является совместимым, будет выброшен `ClassCastException`.

Конвертация объектных ссылок

```
Grapefruit g, g1;
Squeezable s;
g = new Grapefruit();
s = g;          // Convert Grapefruit to
Squeezable (OK)
g1 = s;         // Convert Squeezable to
Grapefruit (Compile error)
```

Модуль 7 Расширенные вопросы

- Вложенные классы
- Статические вложенные классы
- Классы, определенные внутри метода
- Анонимные классы
- Конвертация объектных ссылок при присваивании, вызове метода
- Конвертация и приведение объектных ссылок
- **Assertions**

Assertions

- Assert – механизм отладки приложения, введенный в Java 1.4, позволяющий проверить правильное выполнение метода (предусловие, постусловие и инвариант).

- Предусловие – это ограничение, которое должно быть удовлетворено на входе в метод.

- Постусловие – ограничени, которое должно быть выполнено при выходе из метода.

- Инвариант – ограничение на состояние класса, которое должно выполняться при входе и выходе из любого неприватного метода.

Assertions

- Ключевое слово **assert** имеет следующий синтакс:
  - `assert Expression1;`
  - `assert Expression1:Expression2;`
- **Expession1** должно быть типа **boolean**. Если оно вычисляется как **true**, ничего не происходит, иначе ошибка **AssertionError**.
- Если **Expression2** определено, то в конструктор **AssertionError** передается **Expression2.toString()**.

Внимание! По умолчанию **assertions** отключены. Для включения нужно использовать флаг **-ea**.

Assertions

```
private Book reserveACopy(String title,
Member member) {
   assert isValidTitle(title);
   Book book = getAvailableCopy(title);
   reserve(book, member);

   assert bookIsInStock(book);
   return book;
  }
```

- Предусловие проверяется с помощью механизма **asserts** только в приватном методе.

Проверка предусловия public метода

```
public Book reserveACopy(String title,
Member member) {
     if (!isValidTitle(title))
        throw new
IllegalArgumentException("Bad title: " +
title);
     Book book =
getAvailableCopy(title);
     reserve(book, member);
     assert bookIsInStock(book);
     return book;
}
```

Модуль 7 Расширенные вопросы

- Вложенные классы
- Статические вложенные классы
- Классы, определенные внутри метода
- Анонимные классы
- Конвертация объектных ссылок при присваивании, вызове метода
- Конвертация и приведение объектных ссылок
- Assertions