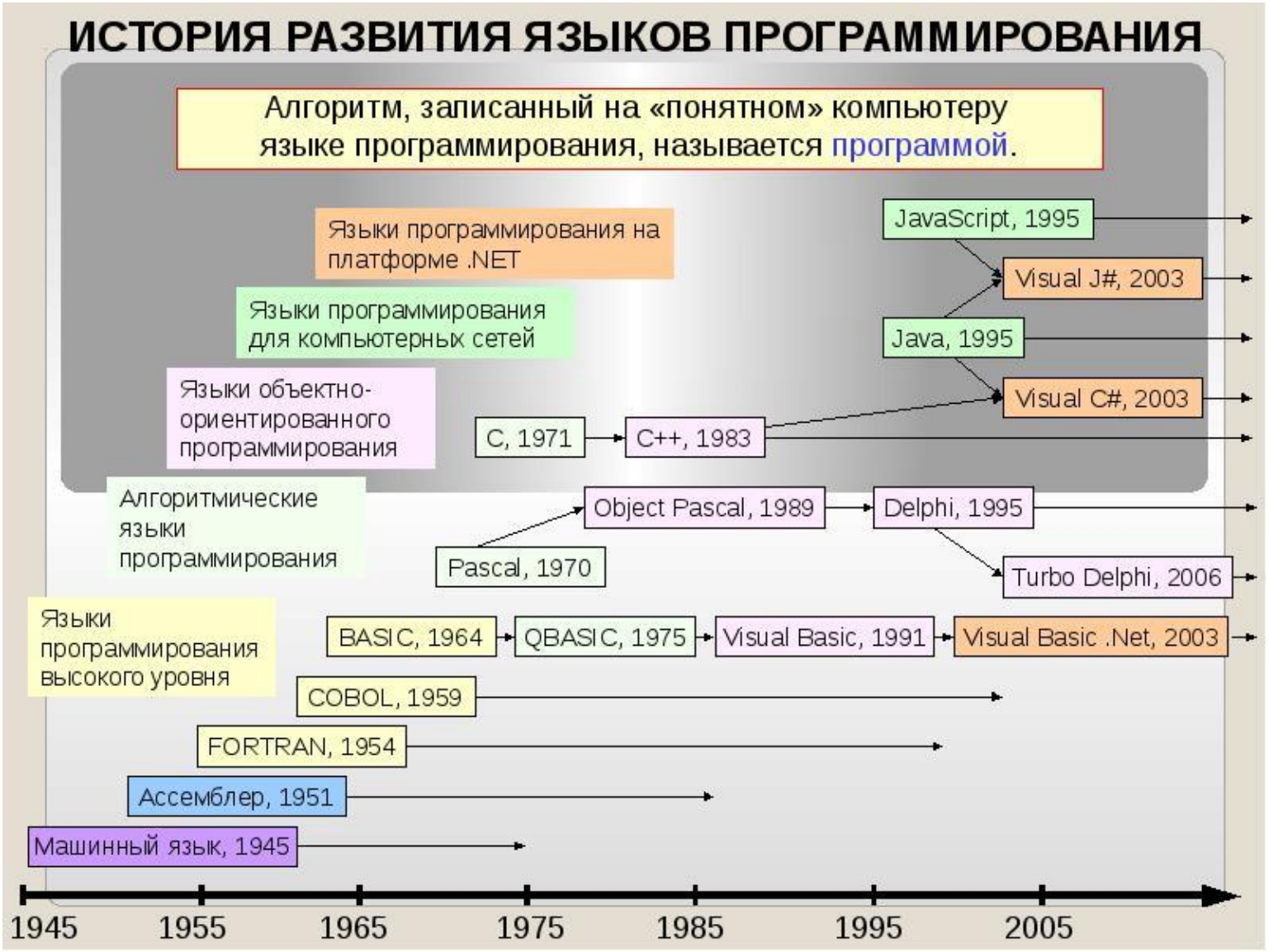
Прикладне програмування

Викладач: Мамчур Дмитро Григорович

Структура курсу

Лекції:
17
Лабораторні:
18
Форма контролю:
Іспит

Вступ


ИСТОРИЯ РАЗВИТИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ
Алгоритм, записанный на «понятном» компьютеру языке программирования, называется программой.
Языки программирования на платформе .NET
Языки программирования для компьютерных сетей
Языки объектно-ориентированного программирования
Алгоритмические языки программирования
Языки программирования высокого уровня
JavaScript, 1995
Visual J#, 2003
Java, 1995
Visual C#, 2003
C, 1971
C++, 1983
Object Pascal, 1989
Delphi, 1995
Pascal, 1970
Turbo Delphi, 2006
BASIC, 1964
QBASIC, 1975
Visual Basic, 1991
Visual Basic .Net, 2003
COBOL, 1959
FORTRAN, 1954
Ассемблер, 1951
Машинный язык, 1945
1945
1955
1965
1975
1985
1995
2005

Вступ


Языки программирования
Низкого уровня
Высокого уровня
Императивные
Декларативные
Процедурные
Об.-ориентирован.
Функциональные
Логические

Вступ

Java - кроссплатформенный, объектно-ориентированный, бесплатный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle).

Основные достоинства языка

- Наибольшая среди всех языков программирования степень переносимости программ.
- Мощные стандартные библиотеки.
- Встроенная поддержка работы в сетях (как локальных, так и Internet/Intranet).

Основные недостатки

- Низкое, в сравнении с другими языками, быстродействие, повышенные требования к объему оперативной памяти (ОП).
- Большой объем стандартных библиотек и технологий создает сложности в изучении языка.
- Постоянное развитие языка вызывает наличие как устаревших, так и новых средств, имеющих одно и то же функциональное назначение.

Основные особенности

- Java является полностью объектно-ориентированным языком. Например, C++ тоже является объектно-ориентированным, но в нем есть возможность писать программы не в объектно-ориентированном стиле, а в Java так нельзя.
- Реализован с использованием интерпретации P-кода (байт-кода). Т.е. программа сначала транслируется в машинонезависимый P-код, а потом интерпретируется некоторой программой-интерпретатором (виртуальная Java-машина, JVM).

Вступ

Создатель Java – Джеймс Гослинг (США)

- Первое применение – бытовая электроника (микроволновые печи, стиральные машины, пульты управления)

Первое название языка – Oak («Дуб»)

- В честь дуба, стоявшего напротив офиса Джеймса Гослинга
- К тому времени уже был ещё один язык Oak

Название Java произошло от сорта кофе

- Это кофе производится на о. Ява (Индонезия)
- Его очень часто употреблял и первые разработчики языка

Duke - талисман языка Java

- Ежегодно проводится конкурс Duke Choice Awards
- В 2011 году Duke изменил свой внешний вид

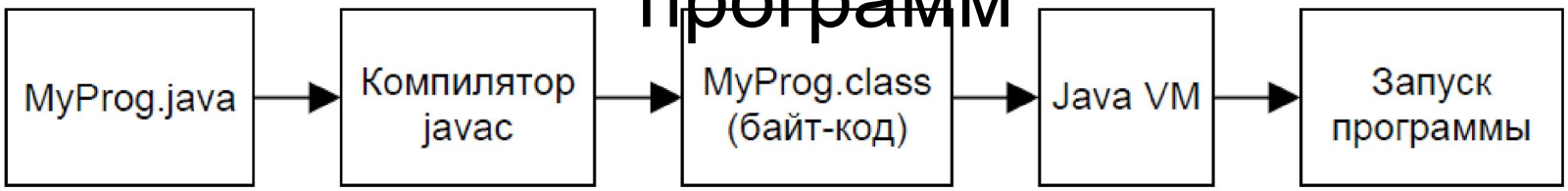
http://www.tiobe.com

Position Aug 2013	Position Aug 2012	Delta in Position	Programming Language
1	2	↑	Java
2	1	↓	C
3	4	↑	C++
4	3	↓	Objective-C
5	6	↑	PHP
6	5	↓	C#
7	7	=	(Visual) Basic
8	8	=	Python
9	11	↑↑	JavaScript
10	10	=	Ruby

Особенности Java

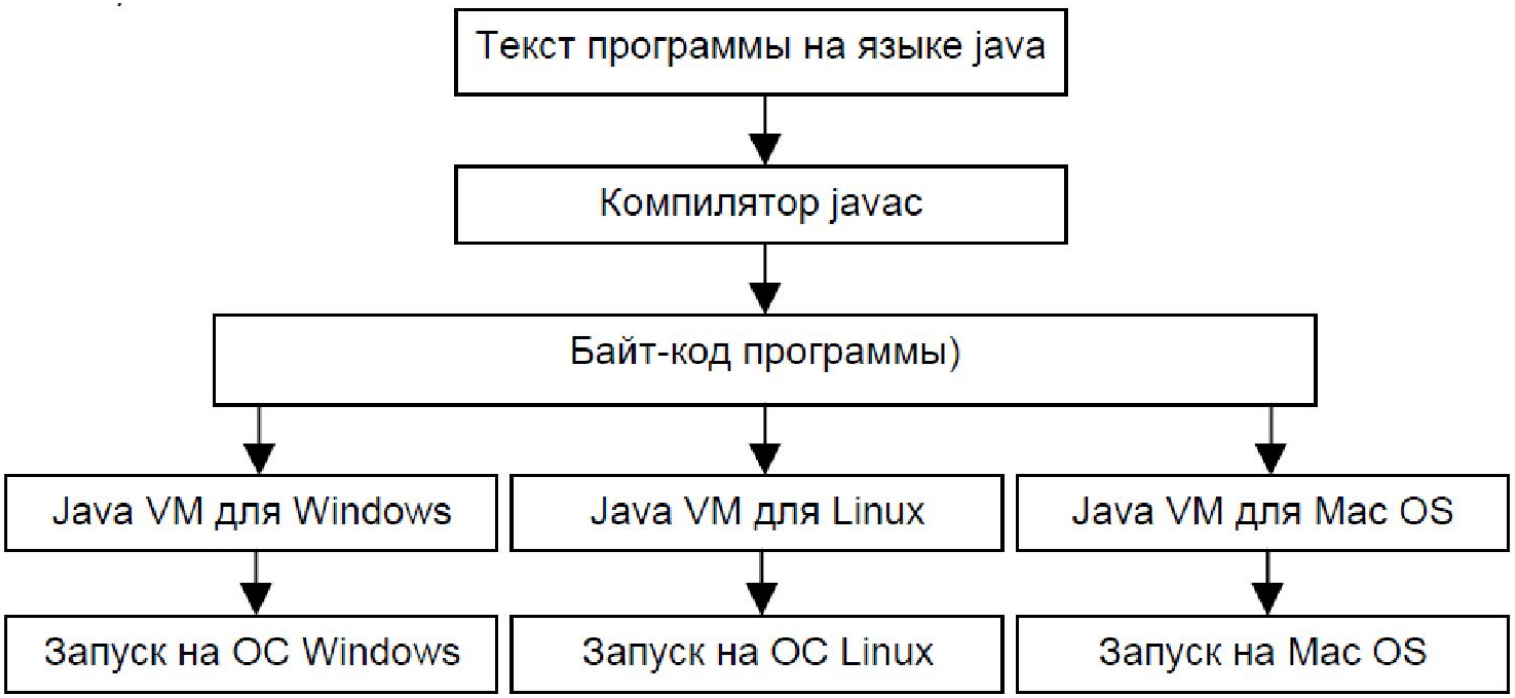
- кроссплатформенность
- объектная ориентированность
- привычный синтаксис C/C++
- безопасность
- ориентация на Internet
- динамичность
- простота освоения

Особенности функционирования Java-программ



Жизненный цикл разработки и запуска программы на языке Java

Файлы байткода исполняются виртуальной машиной Java (Java VM). Для каждой операционной системы или устройства разрабатывается своя Java VM, в то время как байт-код программы остается неизменным



Средства разработки и выполнения Java

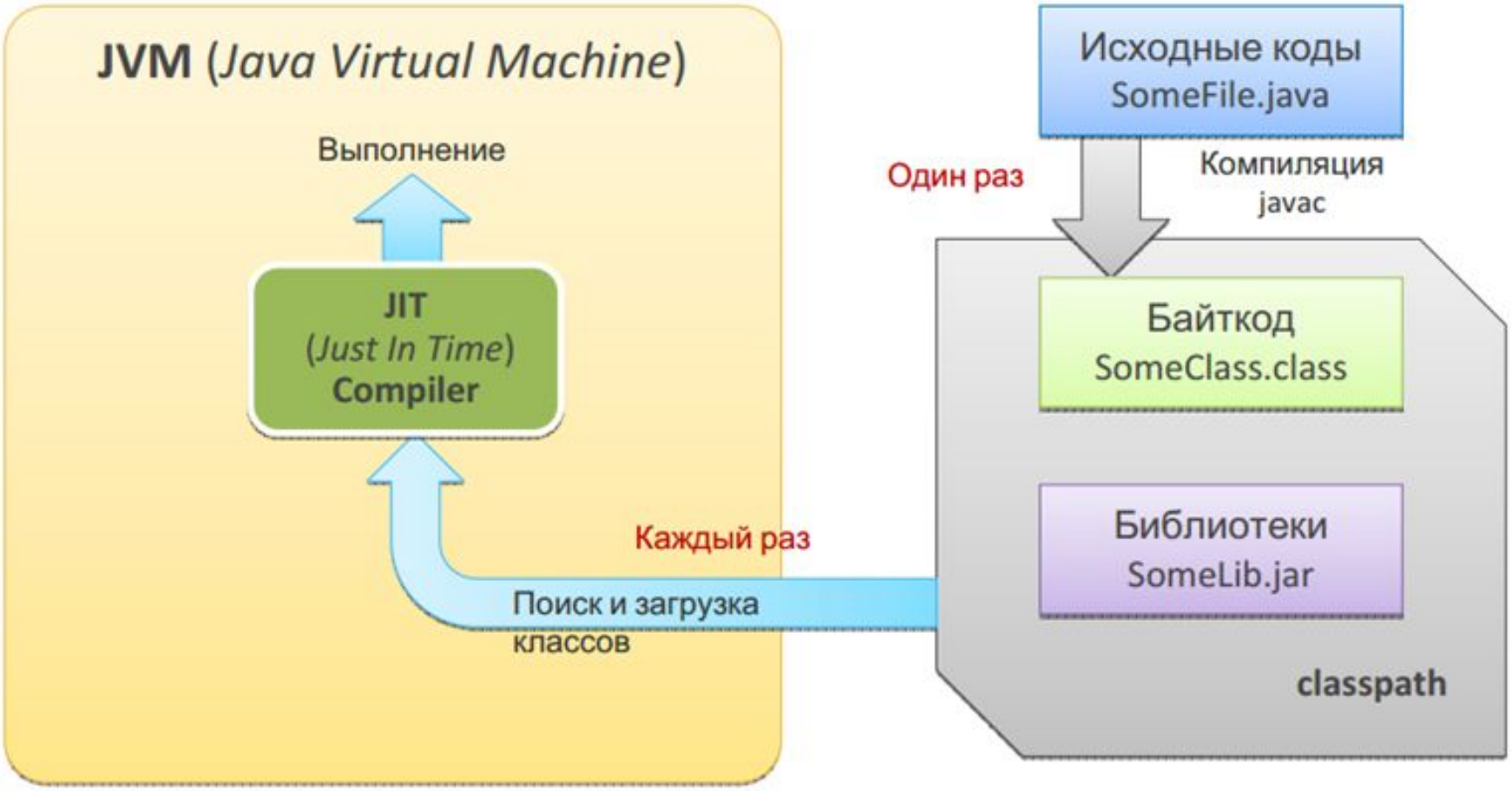
- **Java Runtime Environment, JRE** – это исполнительная среда Java, в которой выполняются программы, написанные на этом языке. Среда состоит из виртуальной машины – **Java Virtual Machine(JVM)** и библиотеки Java-классов. JRE является частью JDK.
- **Java Virtual Machine, JVM** – это виртуальная машина Java — основная часть исполняющей среды **JRE**. Виртуальная машина Java интерпретирует и исполняет байт-код Java. Байт-код получают посредством компиляции исходного кода программы с помощью компилятора Java (стандартный - javac).
- **Java Development Kit**, **JDK** – это бесплатно распространяемый корпорацией Sun комплект разработчика приложений на языке Java, включающий в себя компилятор Java (javac), стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему Java (JRE). В состав JDK не входит интегрированная среда разработки на Java (IDE), поэтому разработчик, использующий только JDK, вынужден использовать внешний текстовый редактор и компилировать свои программы, используя утилиты командной строки.
- **Java 2 Standart Edition**, **J2SE** – это стандартная редакция языка Java, используемая для разработки простых Java-приложений. Используя данную редакцию можно создавать апплеты, консольные приложения, приложения с графическим интерфейсом пользователя.
- **Java 2 Enterprise Edition**, **J2EE** – это редакция языка Java для разработки распределенных приложений масштаба предприятия. Включает в себя технологию Enterprise Java Beans (EJB), Java Server Pages (JSP) и сервлеты(Servlets). Каждая из этих технологии, в свою очередь также имеет свой отдельный номер версии..
- **Java 2 Micro Edition**, **J2ME** – это редакция языка Java для разработки приложений для микрокомпьютеров (мобильных устройств). В нее входят "облегченные" стандартные классы и классы для написания мидлетов (Midlets). Мидлеты – это аналоги аплетов, но только приспособленные специально для небольших устройств. В них также поддерживается графика, звук, реакция на события (нажатие кнопок и т.д.). Java ME наиболее полно соответствует начальному предназначению Java – платформы для написания программ для бытовых устройств.

Среда выполнения и разработки Java

JRE (*Java Runtime Environment*)
программное обеспечение, необходимое для запуска приложений, созданных с помощью Java. Состоит из виртуальной машины — Java Virtual Machine и библиотеки Java-классов

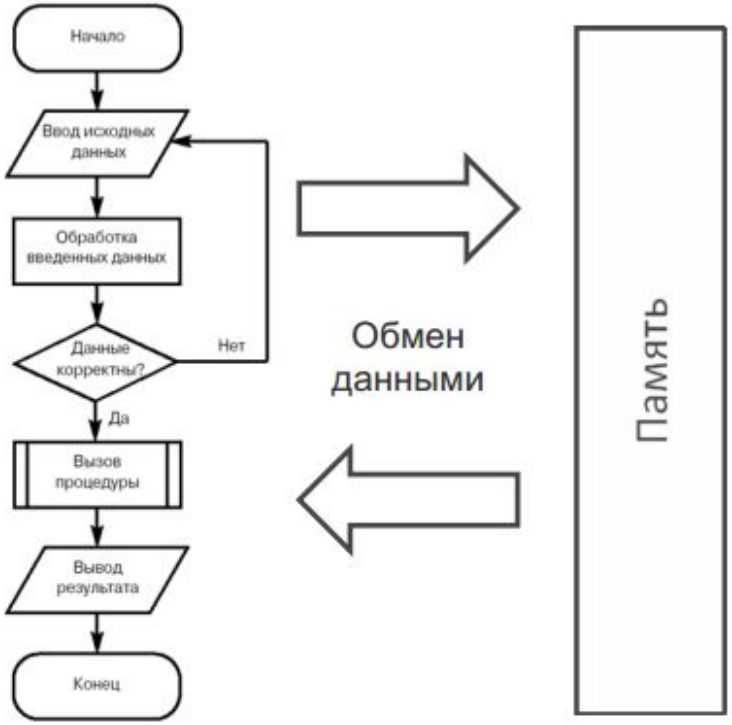
JDK (*Java Development Kit*)
комплект разработчика приложений на языке Java, включающий в себя компилятор Java (javac), стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему Java (JRE).

Среда выполнения и разработки Java



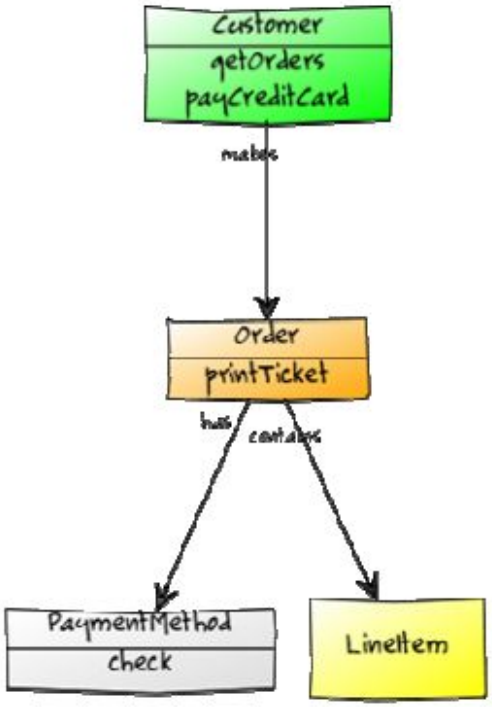
Среда выполнения и разработки Java

Процедурный подход



Программа – алгоритм последовательного вызова проедур изменения данных в памяти.

Объектно-ориентированный



Программа – взаимодействие объектов, компонентов, отсылка и обработка событий.

Java и объектно-ориентированное программирование

Достоинства

- Конструирование из простых компонент (абстракция).
- Данные связаны с операциями обработки.
- Инкапсуляция делает код более безопасным.
- Повторное использование компонент.
- Обобщенные алгоритмы.
- Изменение поведения во время выполнения (полиморфизм).
- Создание полуфабрикатов или фреймворков.

Недостатки

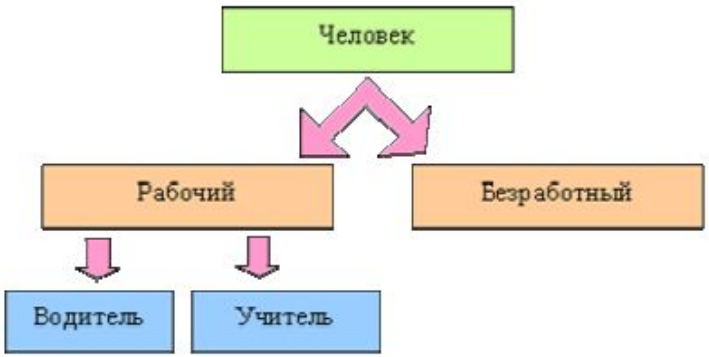
- Необходимость изучения концепций ООП
- Обилие библиотек компонентов повторного использования.
- Проектирование классов сложный процесс.
- Меньшее быстродействие
- Больший расход памяти
- Излишняя универсальность

Java и объектно-ориентированное программирование

Абстракция позволяет акцентировать внимание на способах использования объекта и не вдаваться в подробности его реализации.

•**Инкапсуляция** — свойство языка программирования, позволяющее объединить и защитить данные и код в объекте и скрыть реализацию объекта от пользователя (прикладного программиста). При этом пользователю предоставляется только спецификация (интерфейс) объекта.

Java и объектно-ориентированное программирование



- **Наследование** – описание нового класса на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.
  - Позволяет избавиться от дублирования кода.
  - Позволяет добавить новую функциональность в класс.
  - Позволяет описать отношения обобщения

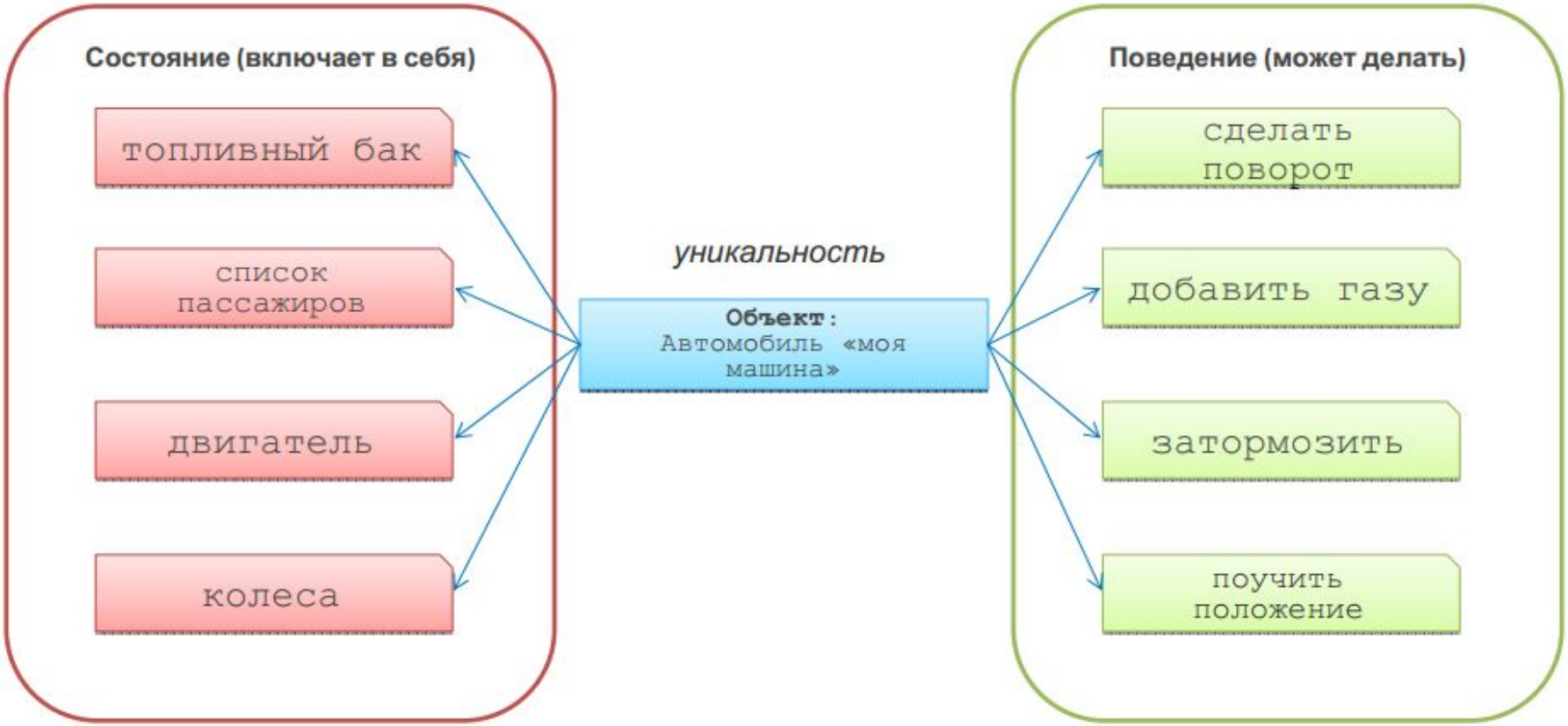
```
class Грузовик extends Автомобиль {
    Кузов кузов;
    процедура загрузить(груз) {
        кузов.загрузить(груз);
        кузов.проверитьПерегруз();
    }
}

Грузовик мойГрузовик = new Грузовик();
мойГрузовик.загрузить(помидоры);
мойГрузовик.добавитьГазу(немного);
```

Java и объектно-ориентированное программирование

- **Полиморфизм** — возможность объектов с одинаковой спецификацией иметь различную реализацию. При этом различные объекты могут быть использованы одинаковым образом.
  - Позволяет писать более абстрактные программы.
  - Позволяет усилить повторное использование кода.
- Реализуется с помощью наследования и интерфейсов.

```
class Автостоянка {
    СписокАвтомобилей автомобили;
    процедура добавить(автомобиль) {
        автомобиль.закрыть();
        автомобиль.включитьСигнализацию();
        автомобили.добавить(автомобиль);
    }
}
Автостоянка стоянка = дом.гдеБлижайшаяАвтостоянка()
стоянка.добавить(мойАвтомобиль);
стоянка.добавить(мойГрузовик);
```

Java и объектно-ориентированное программирование



Java и объектно-ориентированное программирование


```
class Автомобиль {
    ТопливныйБак бак;
    СписокПассажиров пассажиры;
    Колеса колеса;
    Двигатель двигатель;

    процедура сделатьПоворот(угол);
    процедура добавитьГазу(уровень);
    процедура затормозить();
    Координаты получитьПоложение();
}
```

```
class Двигатель {
    Свечи свечи;
    Цилиндры цилиндры;
    Карбюратор карбюратор;

    процедура увеличитьОбороты();
    процедура переключитьРежим(режим);
}
```

```
Автомобиль мояМашина = я.купитьМашину();
Двигатель двигательМоейМашины = мояМашина.двигатель;
двигательМоейМашины.переключитьРежим(форсированный);
```

Java и объектно-ориентированное программирование

Объект –
экземпляр класса.

- состояние
- поведение
- уникальность

Класс –
тип объекта.

- определяет набор свойств и интерфейс взаимодействия.
- определяет поведение (реализацию)

```
Автомобиль мой = new Автомобиль();
Автомобиль жены = new Автомобиль();
Топливо топливо = жены.слитьТопливо();
мой.заправить(топливо);
мой.двигаться();
```

```
class Автомобиль {
    Двигатель двигатель;
    процедура двигаться() {
        двигатель.завести();
        двигатель.добавитьГазу();
    }
}
```

Java и объектно-ориентированное программирование

Основная конструкция языка программирования Java, основной объект, с которым можно что-то делать – это **класс**. У каждого класса есть какие-то характеристики, называемые **полями** (другими словами – переменные) и умения что-то делать, называемые **методами** (другими словами - функции).

```
public class wasinkremenchuk{
    public static void main(String[] args) {
    int n=5;
    System.out.println("I was in Kremenchuk " + n + "
times!");
    }
}
```

В приведенной программе `wasinkremenchuk` – это класс, `main` – это метод, `n` – поле.

Java и объектно-ориентированное программирование

В каждой программе, которую мы собираемся запускать на выполнение, должен быть метод main. Этот метод будет выполнен при запуске программы.

Объявление класса {
    Объявление полей...
    Описание методов...
    Описание метода `main`
}

Java и объектно-ориентированное программирование

Покажем теперь, как класс из одной программы можно использовать в другой программе. Ниже приведен текст программы krem.java.

```
public class krem {
    public void reklama(String NapravlenieObuchenija, int
    ChisloBudzhetnyhmest) {
        System.out.println("В КрНУ около 5 000 студентов
        учатся по более чем 30 направлениям обучения.");
        System.out.println("Одним из этих направлений
        является направление "+NapravlenieObuchenija);
        System.out.println("Число бюджетных мест по этому
        направлению: "+ChisloBudzhetnyhmest);
    }
}


public class telek{
    public static void main(String[] args) {
        krem k=new krem();
        k.reklama("Компьютерная инженерия",30);
    }
}
```

Java и объектно-ориентированное программирование

Чтобы использовать методы класса krem, нужно создать **экземпляр** этого класса. Это делается в строке:

```
krem k = new krem();
```

При этом создается экземпляр `k` класса `krem`.

В общем виде создание экземпляра заданного класса выглядит так:

```
имя_класса имя_переменной = new имя_класса()
```

Ключевое слово `new` как раз и говорит виртуальной машине Java о том, что в памяти выделяется место под новый экземпляр класса.

Теперь с помощью `k` можно вызывать метод `reklama`: `k.reklama`