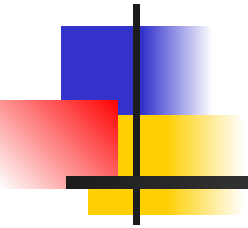
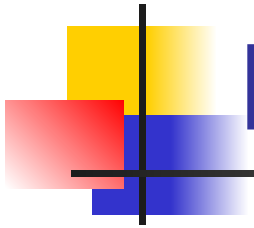


Занятие 1

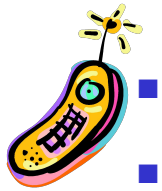
Введение в язык программирования Java



Цели

- Описать историю языка Java
- Кратко объяснить, что такое Java
- Перечислить типы программ Java
- Перечислить возможности Java
- Объяснить различие между апплетами и приложениями
- Описать виртуальную машину Java Virtual Machine (JVM)
- Перечислить функциональные возможности некоторых сред разработки (IDE)
- Ознакомиться с пакетом JDK и содержащимися в нём средствами
- Проанализировать перспективные пути развития и технологии

Введение в язык программирования Java



- В 1990 г. фирма Sun Microsystems начали разработки языка ‘Oak’
- 1994 г. – Sun выпустила браузер HotJava.
- Вопросы переносимости и независимости от платформы являлись главными проблемами для пользователей Интернета.
- В 1995 г. язык был переименован в Java.
- Java, являясь безопасным, переносимым и независимым от платформы языком, продемонстрировал способность решать крупномасштабные проблемы во всей сети Интернет.

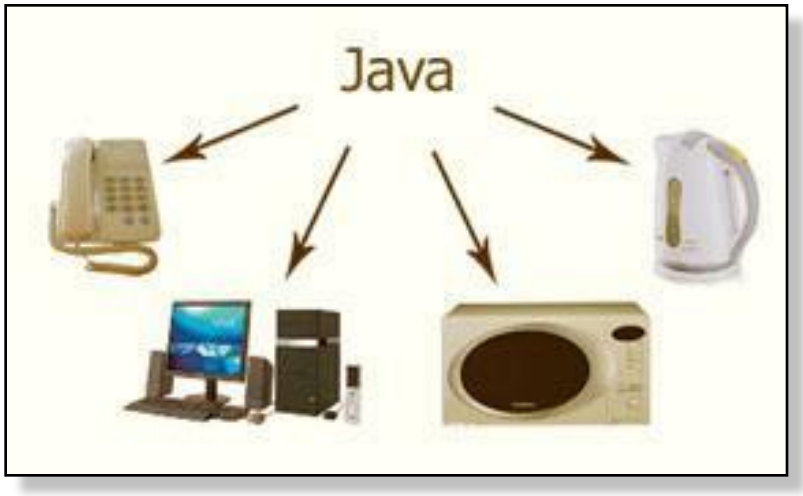
Что такое Java?

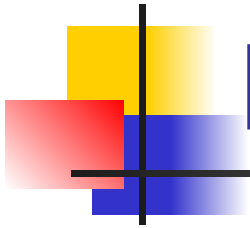
- Объектно-ориентированный язык программирования (компилируемый и интерпретируемый).
- Его преимущество для пользователей World Wide Web заключается в появлении безопасных, платформонезависимых приложений, которые могут использоваться в любом узле Internet.
- Программисты, создающие приложения на Java, пишут свой код всего один раз - им не приходится «переносить» свои приложения на все возможные программные и аппаратные платформы.

Что такое Java?

Сначала Java (официальный день рождения технологии Java — 23 мая 1995 г.) предназначалась для программирования бытовых электронных устройств (Представлен фирмой Sun Microsystems)

Потом Java стала применяться для программирования браузеров - появились апплеты. Затем оказалось, что на Java можно создавать полноценные приложения.



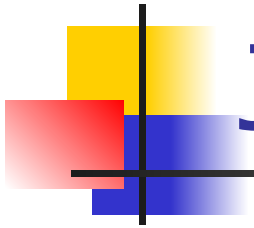


История Java

Далее появились сервлеты и EJB (Enterprise JavaBeans).

Серверы должны взаимодействовать с базами данных — появились драйверы JDBC (Java DataBase Connection).

Взаимодействие оказалось удачным, и многие системы управления базами данных и даже операционные системы включили, Java в свое ядро, например Oracle, Linux, MacOS.



Java и сеть Интернет

- Многие рассматривают Java в первую очередь как средство создания апплетов для World Wide Web.
- Термином «апплет» в Java обозначается мини-приложение, работающее внутри Web-страницы.
- После того как апплет загружен на компьютер пользователя, он может выполнять определенные задачи и взаимодействовать с пользователем через браузер, не требуя ресурсов Web-сервера.

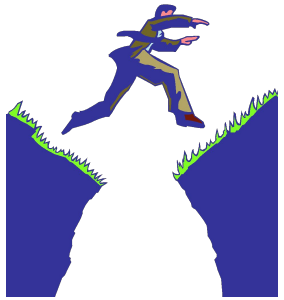
Java и сеть Интернет

- Программы в сети являются либо статическими, либо динамическими.
- Апплеты помогают разрабатывать динамические программы.
- Апплеты работают в web-браузерах, совместимых с Java.

- Апплеты могут активно реагировать («отвечать») на действия пользователя и вводимые им данные.

Вопросы безопасности

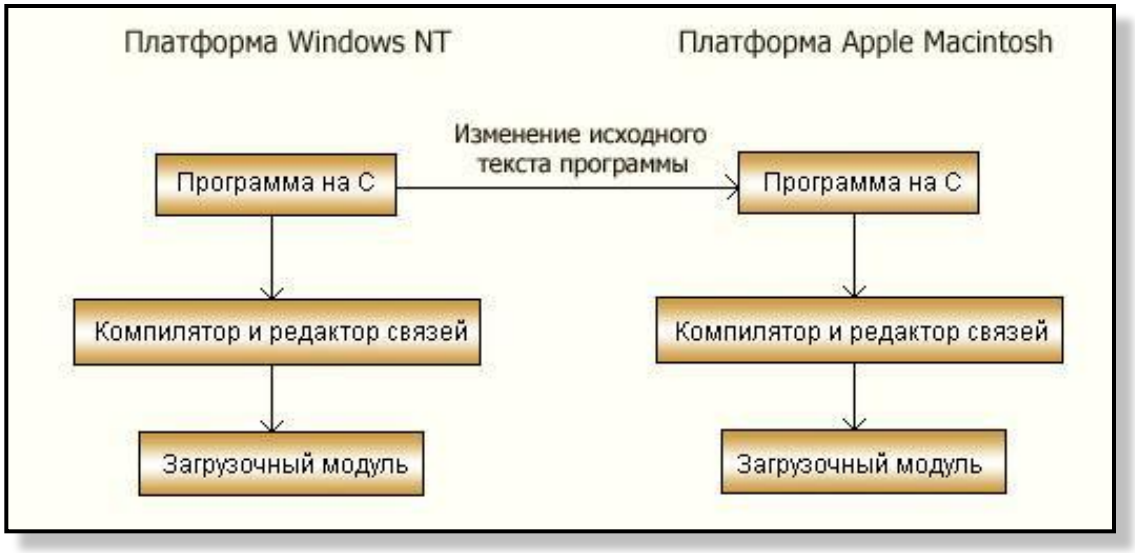
- Апплет должен быть загружен в системе пользователя, прежде чем он сможет начать работу.
- Это создаёт потенциальный риск для системы пользователя.

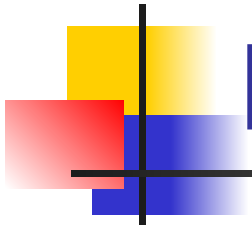


- Поэтому апплетам запрещён доступ ко всем разделам жёсткого диска.

Мобильность Java

- Независимость от платформы означает легкость переноса программы с одного компьютера на другой без каких-либо трудностей.





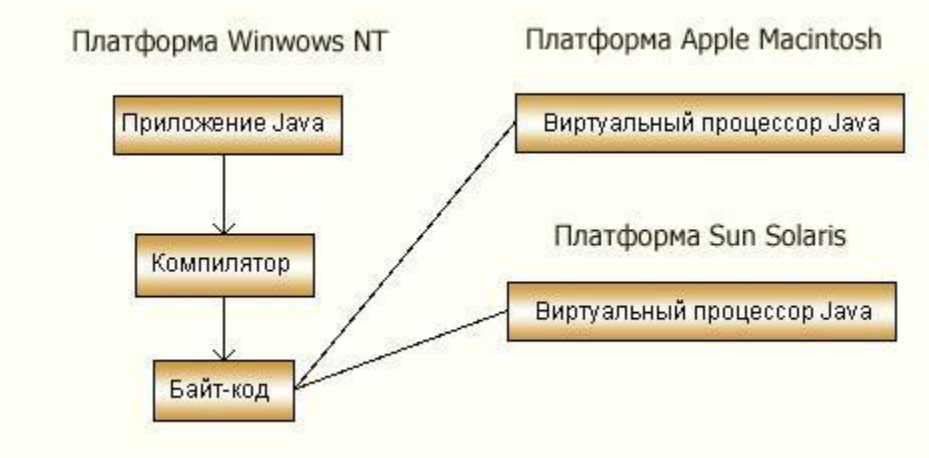
Мобильность Java

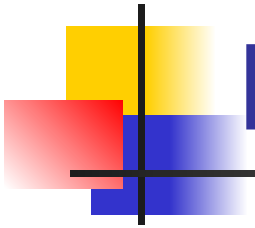
- Программа на языке Java компилируется в двоичный модуль, состоящий из команд виртуального процессора Java.
- Такой модуль содержит байт-код, предназначенный для выполнения Java-интерпретатором.
- На каждой платформе используется свой интерпретатор, или, точнее говоря, свой виртуальный процессор Java.

Мобильность Java

- Таким образом, приложение Java компилируется и отлаживается только один раз.
- Приложение Java не обращается напрямую к интерфейсу операционной системы. Вместо этого оно пользуется готовыми стандартными библиотеками классов, содержащими все необходимое для организации пользовательского интерфейса.

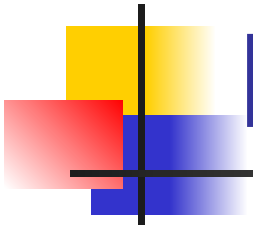
Внутренняя реализация библиотек классов, разумеется, зависит от платформы. Все загрузочные модули, реализующие возможности этих библиотек, поставляются в готовом виде вместе с виртуальной машиной Java.





Выполнение Java-программы

- Программа, написанная на одном из языков высокого уровня, к которым относится и язык Java, так называемый **исходный модуль**, не может быть сразу же выполнена.
- Ее сначала надо откомпилировать, т. е. перевести в последовательность машинных команд — **объектный модуль**.
- Но и он, как правило, не может быть сразу же выполнен: объектный модуль надо еще скомпоновать с библиотеками использованных в модуле функций и разрешить перекрестные ссылки между секциями объектного модуля, получив в результате **загрузочный модуль** — полностью готовую к выполнению программу.
- Исходный модуль, написанный на Java, не может избежать этих процедур, но здесь проявляется главная особенность технологии Java — программа компилируется сразу в машинные команды, но не команды какого-то конкретного процессора, а в команды так называемой виртуальной машины Java (JVM, Java Virtual Machine).



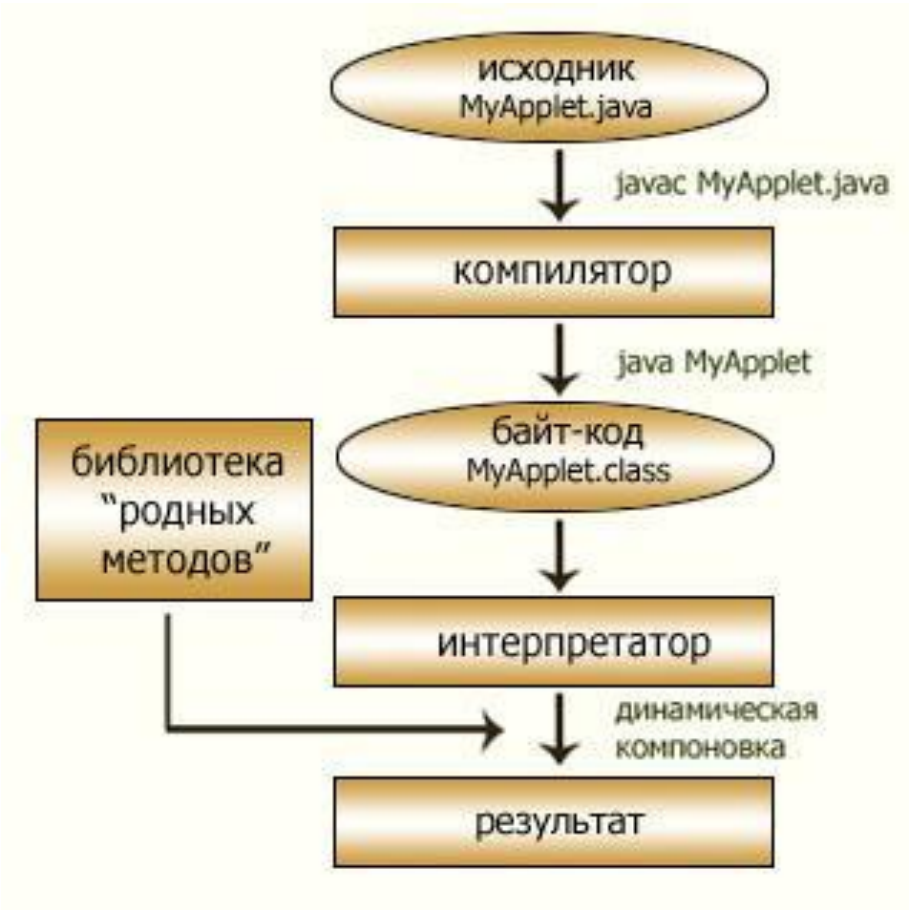
Виртуальная машина Java

- **Виртуальная машина Java** — это совокупность команд вместе с системой их выполнения.
- Виртуальная машина Java полностью стековая, так что не требуется сложная адресация ячеек памяти и большое количество регистров. Поэтому команды JVM короткие, большинство из них имеет длину 1 байт, отчего команды JVM называют **байт-кодами** (bytecodes).
- Полное описание команд и всей архитектуры JVM содержится в спецификации виртуальной машины Java (VMS, Virtual Machine Specification).
- Другая особенность Java — все стандартные функции, вызываемые в программе, подключаются к ней только на этапе выполнения, а не включаются в байт-коды. Происходит **динамическая компоновка** (dynamic binding). Это тоже сильно уменьшает объем откомпилированной программы.

Виртуальная машина Java Virtual Machine (JVM)

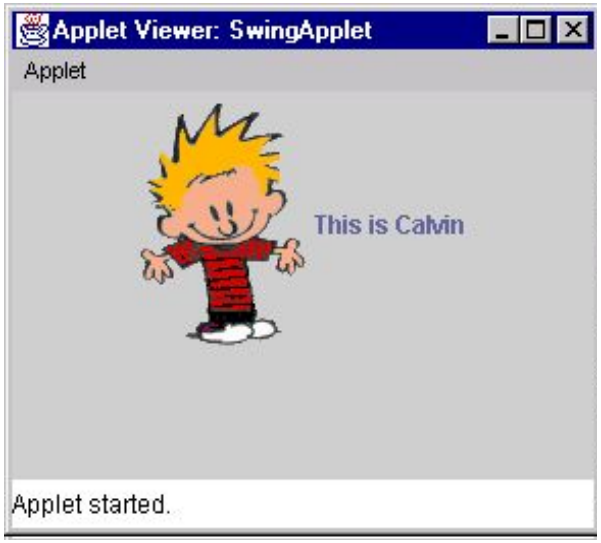
- Содержит компонент интерпретатор, который позволяет осуществлять обмен информацией между байт-кодом Java и операционной системой компьютера.
- Код Java может выполняться на любой платформе с помощью виртуальной машины JVM.
- Обычно JVM считывает и выполняет инструкции Java по одной.
- Виртуальная машина JVM отвечает за обеспечение независимости от платформы и за компактность скомпилированного кода.
- Распознаёт только специальный бинарный формат, называемый файлом класса (class-файл).

Выполнение программы

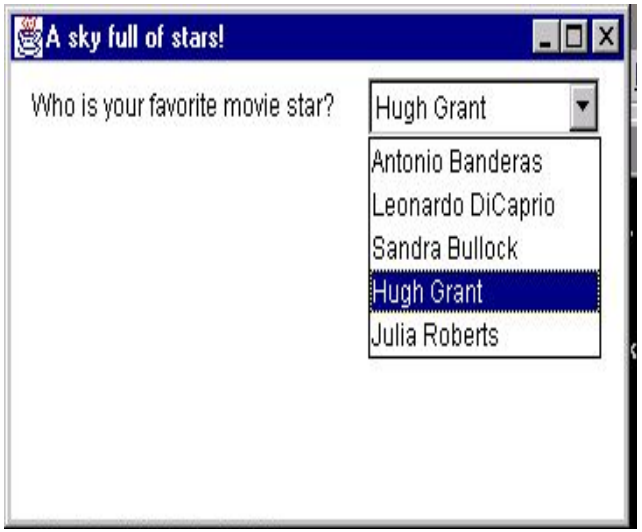


Типы Java-программ

- **Applets - Апплеты**
  - Программы, созданные специально для работы в сети Internet



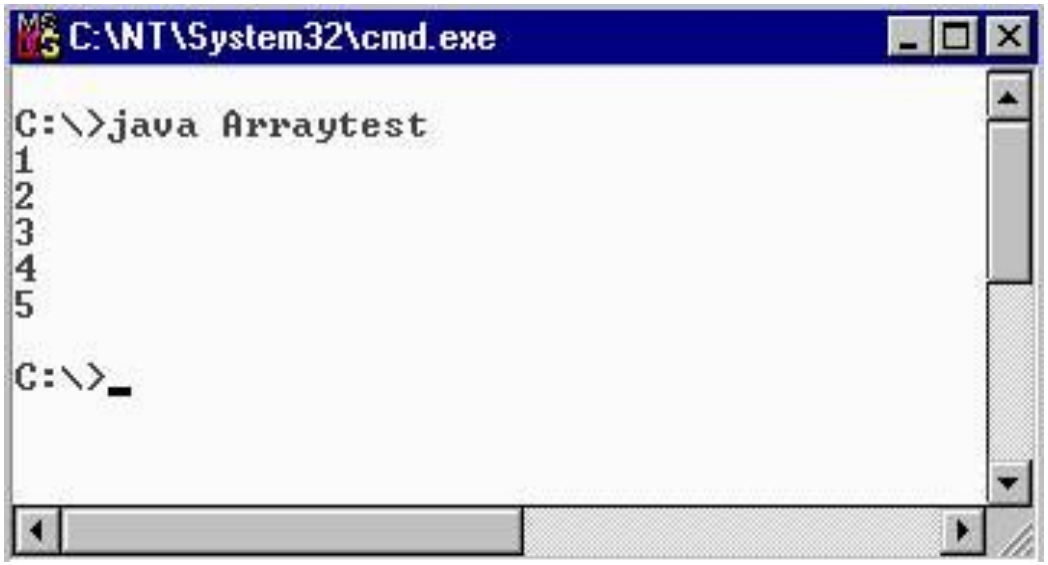
Вывод изображений



Интерфейс GUI для приёма ввода пользователя

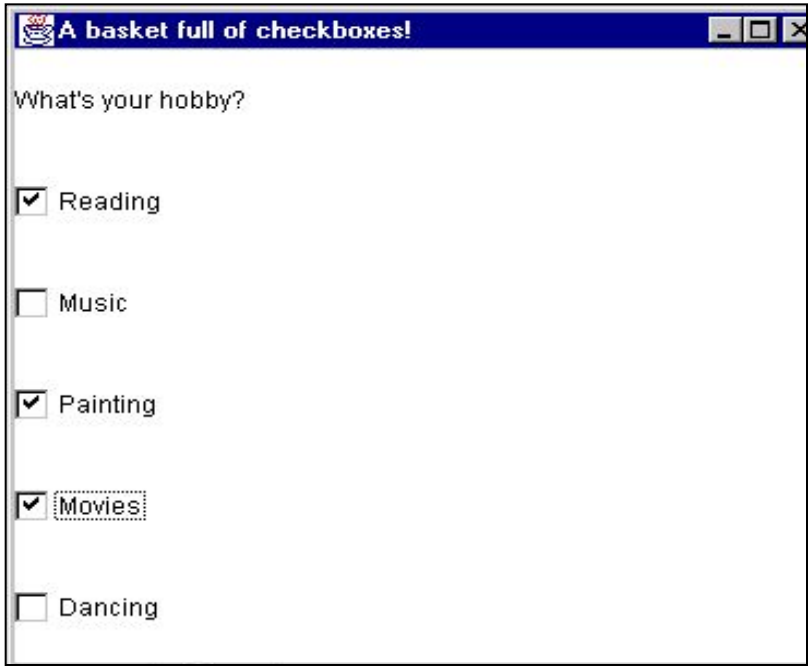
Типы Java-программ

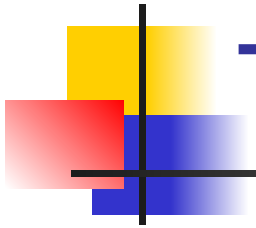
- **Консольные приложения**
  - Java-программы, которые запускаются из командной строки, и не выводят GUI-окон и панелей



Типы Java-программ

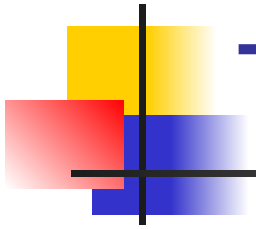
- **GUI-приложения**: Java-программы, которые запускаются и работают независимо и принимают ввод пользователя с использованием средств графического интерфейса GUI (окна, панели и т.д.).





Типы Java-программ

- **Servlets - Сервлеты**
  - Предназначены для web-ориентированной разработки n-звенных приложений.
  - Клиент посылает запрос, который обрабатывается сервером.
  - Программные интерфейсы API стороны сервера расширяют возможности стандартных интерфейсов API и известны под названием Servlets (Сервлеты).
  - Также их называют апплетами стороны сервера.
  - Пример – Обработка HTML-формы, обработка баз данных и выполнение транзакций на стороне сервера.



Типы Java-программ

- **Packages - Пакеты**
  - Библиотеки классов языка Java
  - Программисты могут создавать собственные пакеты или использовать встроенные пакеты
    - `java.awt`, `java.io` и `java.applet` являются примерами пакетов
- **Приложения баз данных**
  - Используют интерфейс JDBC API для соединения с базами данных.

Различия между апплетами и приложениями

Приложения	Апплеты
▪ Для выполнения не требуется браузер ▪ Выполняются под управлением виртуальной машины JVM	▪ Выполняются в java-совместимом web-браузере ▪ Апплеты загружаются и выполняются в web-браузере, поддерживающем Java

Различия между апплетами и приложениями

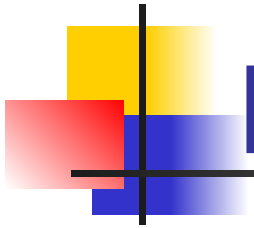
Приложения	Апплеты
▪ Управляют собственным потоком выполнения ▪ Отсутствуют ограничения на чтение или запись в локальной файловой системе	▪ Поток выполнения частично управляется контекстом браузера ▪ Операции чтения/записи в локальной файловой системе запрещены

Сходство апплетов и приложений

- Оба требуют использования стандартных библиотек классов Java.

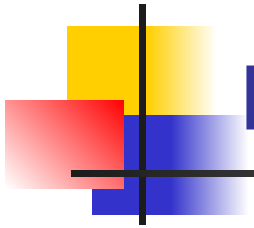
- Оба могут использовать пакет Abstract Windowing Toolkit (AWT).

Средства визуальной разработки



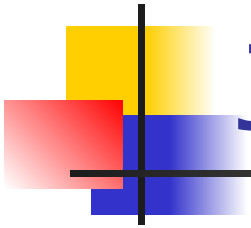
- Помогают быстро и эффективно разрабатывать приложения и апплеты на языке Java.
- Упрощают процесс разработки ПО.
- Включают браузер классов для просмотра и перемещения по иерархии классов.
- Включают редактор исходного кода, который помогает при написании кода программ.

Некоторые средства визуальной разработки



- **Intellij idea** разработанная компанией JetBrains

- **NetBeans IDE**, компанией NetBeans Org
  - Интегрированная среда, предоставляющая возможности визуального проектирования, редактирования, компиляции и отладки для создания приложений



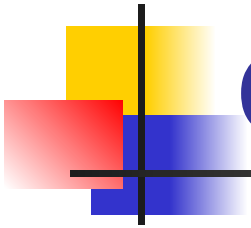
Java Development Kit (JDK)

- Содержит программное обеспечение и инструментальные средства, необходимые для компиляции, отладки и выполнения апплетов и приложений
- Версии:
  - Java 1.0 – первый выпуск
  - Java 1.1 – выпуск 1997 года
  - Java SE 12.0.1 – последний выпуск
- Доступны для бесплатного копирования с официального сайта компании Oracle www.oracle.com/technetwork/java/

Java Development Kit (JDK)

- **JDK (Java Development Kit)** – набор необходимых программных инструментов для полного цикла работы с этим языком программирования: компиляции, интерпретации, отладки, включающий и богатую библиотеку классов.

- Набор программ и классов JDK содержит:
  - компилятор javac из исходного текста в байт-коды;
  - интерпретатор java, содержащий реализацию JVM;
  - программу архивации и сжатия jar;
  - программу сбора документации javadoc;
  - программу serialver, определяющую номер версии класса;
  - библиотеки и заголовочные файлы "родных" методов;
  - библиотеку классов Java API (Application Programming Interface);
  - др.

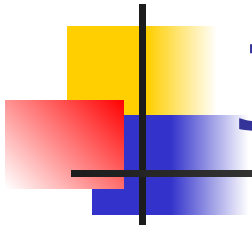


Средства в составе пакета JDK

- **javac**: компилятор, используемый для компиляции исходного кода Java
  - Синтаксис: javac [option] source
  - Файлы исходного кода имеют расширение `.java`
  - Ключи (опции) командной строки

javac MyProgram.java

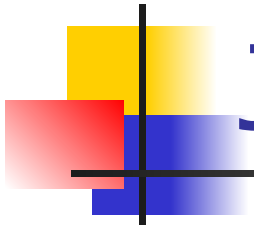
Средства в составе пакета JDK (продолжение)



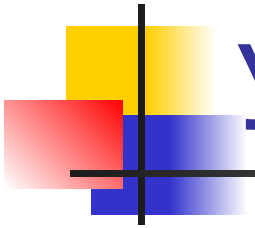
- **java**: интерпретатор, используемый для выполнения байт-кода Java
  - Синтаксис: java [option] classname [arguments]
  - Ключи (опции) командной строки

```
java MyProgram
```

Средства в составе пакета JDK (продолжение)



- **appletviewer**: Используется для просмотра и тестирования апплетов
  - Синтаксис: appletviewer [options] url
- **javadoc**: Документация по инструментальным средствам языка Java
  - Генерирует подробную документацию в формате HTML для любого исходного кода в `.java`–файле или в пакете



Установка JDK

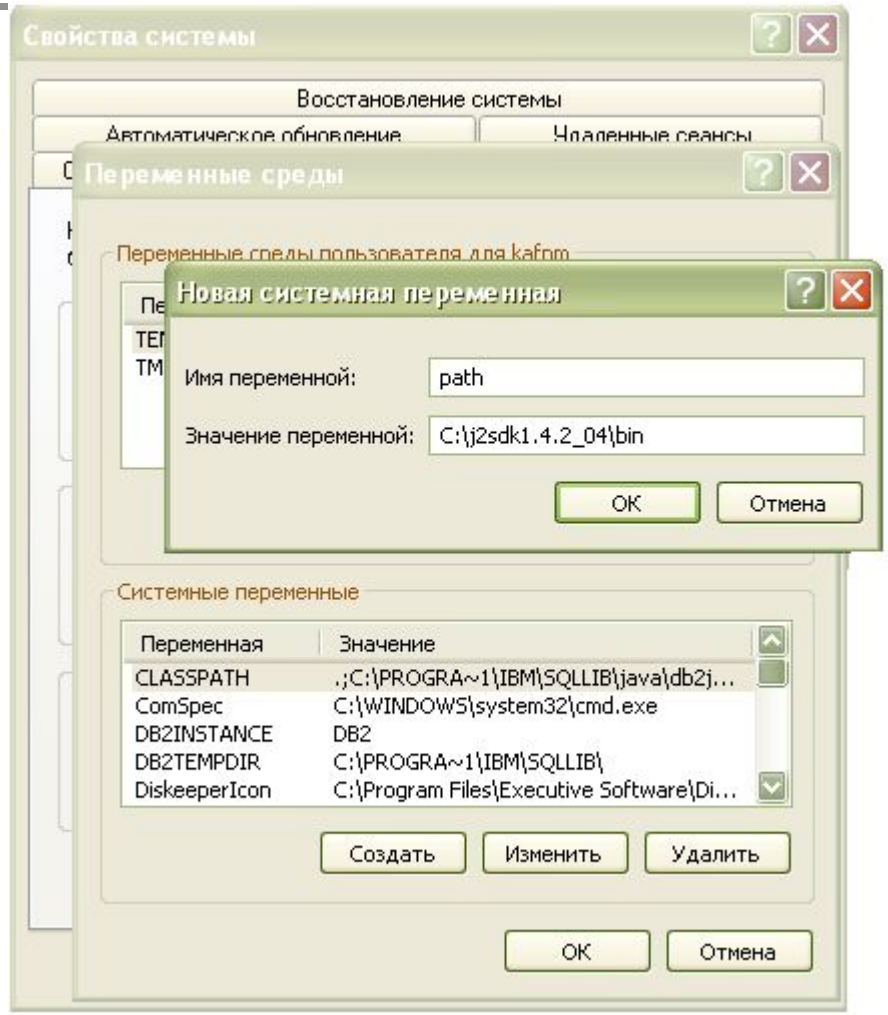
Каталог с названием jdk-12.0.2, а в нем подкаталоги:

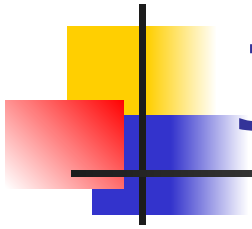
bin, содержащий исполнимые файлы;
jmods, содержит файлы JMOD для стандартной библиотеки;
legal, содержащий лицензии;
docs, содержащий документацию, если вы ее установили;
include, содержащий заголовочные файлы "родных" методов;
lib, содержащий библиотеки классов и файлы свойств;
src, с исходными текстами программ JDK. В новых версиях вместо каталога имеется упакованный файл src.jar.

Установка JDK

Надо определить специальную переменную PATH, содержащую пути к архивным файлам и каталогам с библиотеками классов.

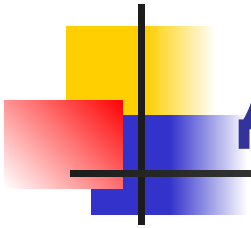
Path= C:\Program Files\Java\jdk-12.0.2\bin





Java Runtime Environment

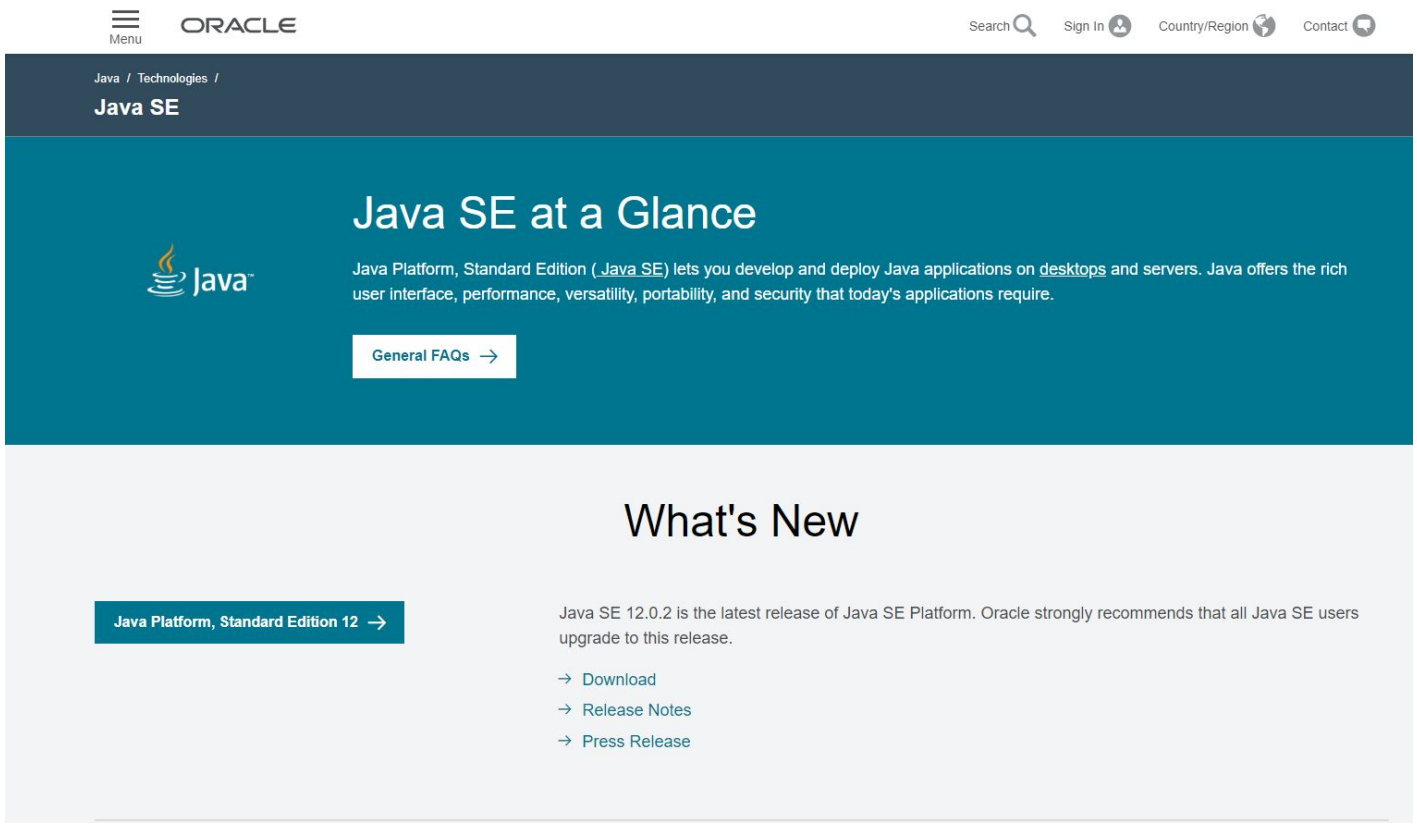
- **JRE (Java Runtime Environment)** -набор программ и пакетов классов JRE содержит все необходимое для выполнения байт-кодов, в том числе интерпретатор java (в прежних версиях облегченнный интерпретатор jre) и библиотеку классов.
- Это часть JDK, не содержащая компиляторы, отладчики и другие средства разработки. Именно JRE или его аналог других фирм содержится в браузерах, умеющих выполнять программы на Java, операционных системах и системах управления базами данных.



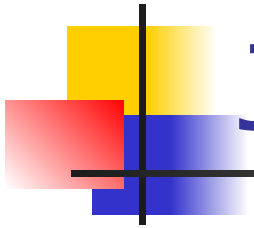
Документация Java

- **Документация Java** – полезный документ, описывающий особенности языка Java и содержащий справку о классах, пакетах, интерфейсах, методах и т.д.
- Скачать документацию с сайта Oracle по адресу https://www.oracle.com/java/technologies/java-se-glance.html

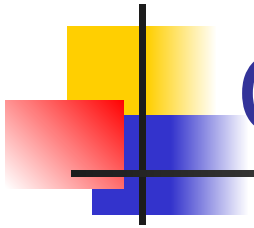
Документация Java



Новые возможности Java 12



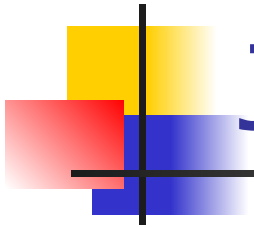
- Улучшены:
  - Оператор switch
  - Улучшения G1
- Microbenchmark Suite – добавляет базовый набор тестов производительности в исходный код JDK. Это облегчает разработчикам запуск и создание тестов производительности.
- JVM Constants API – представляет API для моделирования номинальных описаний ключевых класс-файлов и run-time артефактов.
- Default CDS Archives – ускоряет время сборки JDK путем создания class data-sharing архива.



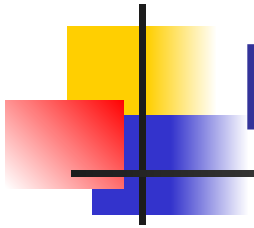
Современные технологии Java

- Создание и развёртывание приложений, которые могут работать в любой операционной системе
- Поддержка распределённой обработки данных в форме таких функциональных возможностей.
- Поддержка управления базами данных в форме JDBC
- Программные компоненты многократного использования в форме JavaBeans.

Современные технологии Java

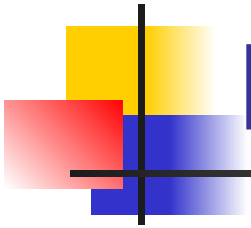


- Корпоративная платформа Java Enterprise Edition (Java EE) включает:
  - EJB - Enterprise JavaBeans — спецификация технологии серверных компонентов, содержащих бизнес-логику
  - JSP - JavaServer Pages — динамическая генерация веб-страниц на стороне сервера
  - Servlet - Обслуживание запросов веб-клиентов



Перспективы развития

- Новые продукты в области технологий, встраиваемых в электронные устройства массового использования с целью обеспечения мобильной обработки данных.
- Миниатюрные устройства, такие как карманные компьютеры и мобильные телефоны, оснащаются такими функциями, как электронная почта, игровые и многи другие возможности.
- Всё это основано на платформе Java Micro Edition (Java ME).



Первая программа на Java

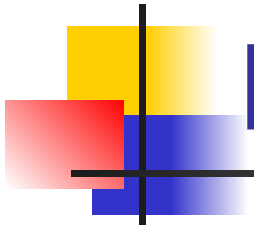
- Реализация первой программы на языке Java выглядит следующим образом:

```
class MyFirstProgramm {
    public static void main(String[] args) {
        System.out.println("My first Java programm!!");
    }
}
```

Первая программа на Java

- Всякая программа представляет собой один или несколько классов, в этом простейшем примере только один класс (class).
- Начало класса отмечается служебным словом class, за которым следует имя класса, выбираемое произвольно, в данном случае MyFirstProgramm.
- Все, что содержится в классе, записывается в фигурных скобках и составляет тело класса (class body).

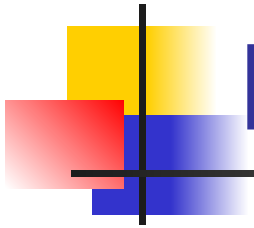
```
class MyFirstProgramm {
    ....
}
```



Первая программа на Java

- Все действия производятся с помощью методов обработки информации, коротко говорят просто метод (method).
- Один из методов обязательно должен называться main, с него начинается выполнение программы.
- В нашей простейшей программе только один метод, а значит, имя ему main. Начало класса отмечается служебным словом class, за которым следует имя класса, выбираемое произвольно, в данном случае MyFirstProgramm.

```
public static void main (String[] args) {

    ...
}
```



Первая программа на Java

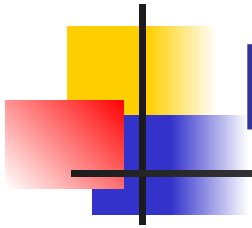
- После имени метода в скобках, через запятую, перечисляются аргументы (arguments) - или параметры метода.
- Для каждого аргумента указывается его тип и, через пробел, имя.
- В примере только один аргумент, его тип — массив, состоящий из строк символов.
- Имя массива может быть произвольным, в примере выбрано имя args.

```
public static void main(String[] args) {

    …
}
```

Первая программа на Java

- Перед типом возвращаемого методом значения могут быть записаны модификаторы (modifiers).
- В примере их два: слово public означает, что этот метод доступен отовсюду; слово static обеспечивает возможность вызова метода main() в самом начале выполнения программы.
- Модификаторы вообще необязательны, но для метода main() они необходимы.

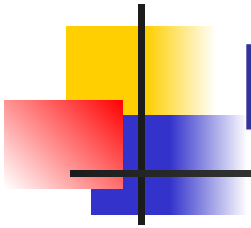
```
public static void main(String[] args) {

    …
}
```



Первая программа на Java

- Единственное действие, которое выполняет метод main() в примере, заключается в вызове другого метода со сложным именем System.out.println и передаче ему на обработку одного аргумента, текстовой константы "My first Java programm!!".
- Текстовые константы записываются в кавычках, которые являются только ограничителями и не входят в состав текста.

```
System.out.println("My first Java programm!!");
```

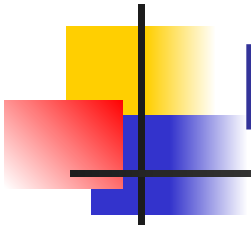


Первая программа на Java

- Составное имя System.out.println означает, что в классе System, входящем в Java API, определяется переменная с именем out, содержащая экземпляры одного из классов Java API, класса PrintStream, в котором есть метод println().

- Действие метода println() заключается в выводе своего аргумента в выходной поток, связанный обычно с выводом на экран текстового терминала, в окно MS-DOS Prompt или Command Prompt или Xterm, в зависимости от системы.

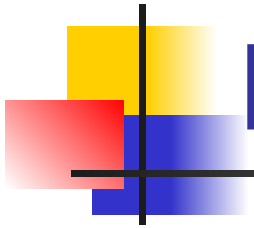
```
System.out.println("My first Java programm!!");
```



Первая программа на Java

- Язык Java различает строчные и прописные буквы, имена main, Main, MAIN различны с "точки зрения" компилятора Java.
- В примере важно писать String, System с заглавной буквы, а main с маленькой.
- Но внутри текстовой константы неважно, писать "MY FIRST JAVA PROGRAMM!!" или "My First Java Programm!!", разница будет видна только на экране.

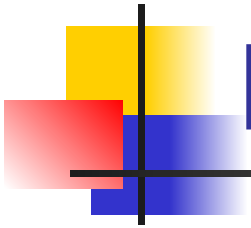
```
System.out.println("My first Java programm!!");
```



Выполнение программы

- Программа написана в каком-либо текстовом редакторе, например, Notepad.
- Теперь ее надо сохранить в файле, имя которого совпадает с именем класса, содержащего метод main(), и дать имени файла расширение java.

MyFirstProgramm.java



Выполнение программы

- Затем вызовем компилятор, передавая ему имя файла в качестве аргумента:

javac MyFirstProgramm.java

- Компилятор создаст файл с байт-кодами, даст ему имя MyFirstProgramm.class и запишет этот файл в текущий каталог.
- Вызов интерпретатора:

java MyFirstProgramm

Результат выполнения