Доля П.Г.
Харьковский Национальный Университет
механико – математический факультет
2015 г.

Начало программирования на Java.

Это небольшое пособие является введением в программирование на Java. Это не учебник, а заметки для тех, кто только начинает знакомиться с этим языком. Здесь вы узнаете как написать, откомпилировать и выполнить вашу первую программу. Для некоторых людей этот первый шаг является психологическим барьером, и мы поможем вам его преодолеть. Конечно, мы напишем не одну программу.

Вы узнаете все, что нужно, чтобы ваши первые программы начали работать, а также опишем некоторые инструменты, которые вы, возможно, пожелаете использовать.

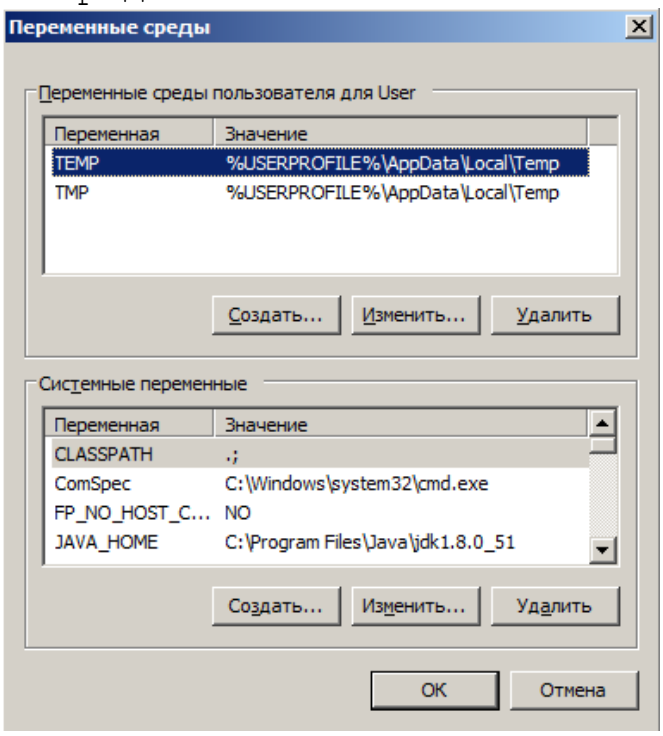
Оглавление

1. Среда программирования JDK.

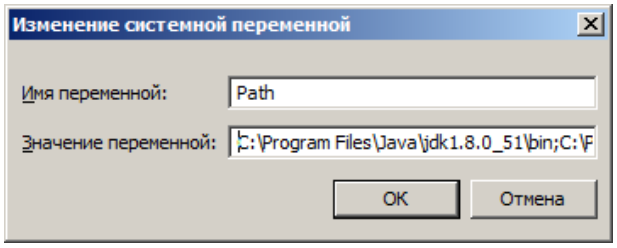
Вначале вы должны установить программную среду на свой компьютер. Полагаем, что вы уже это сделали. Если нет, то лучше всего ее взять с сайта http://www.oracle.com/technetwork/java/javase/downloads/index.html компании Oracle. Выберите на этой странице ссылку JDK и скачайте инсталляционный пакет, соответствующий вашей операционной системе (он бесплатный). Пакет будет называться Java SE Development Kit + номер версии. Например, Java SE Development Kit 8u51. Запустите скачанный файл и следуйте инструкциям мастера установки.

Далее мы будем полагать, что вы работаете в одной из версий ОС Windows. При этом, Java Development Kit (JDK), установленный на ваш компьютер, может работать только в режиме командной строки – компилятор и интерпретатор JAVA программ будут запускаться командами из консольного окна. Позже мы опишем другие способы разработки JAVA программ.

Для правильного функционирования компилятора и интерпретатора Java необходимо сообщить Windows, где расположены файлы программной среды JDK и куда следует помещать результирующие программы. Это делается с помощью системных переменных `PATH`, `CLASSPATH` и `JAVA_HOME`. В Windows 7 для этого откройте `Панель управления->Система` и выберете `Дополнительные параметры системы`. В появившемся окне нажмите кнопку `Переменные среды`.

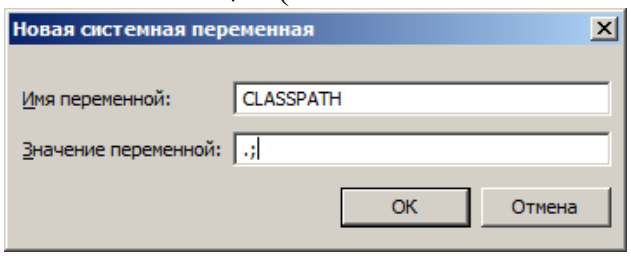


В открывшемся окне в разделе `Системные переменные` нужно выделить имя переменной `PATH` (или `Path`) и нажать кнопку `Изменить`.



Поле `Значение переменной` уже содержит некоторые пути, необходимые для работы других программ. Не испортите их. В начале этого поля перед всеми другими путями введите имя каталога, в котором установлена ваша JDK. Например, C:\Program Files\Java\jdk1.8.0_51\bin и в конце поставьте точку с запятой. Указывать путь надо именно на каталог `...\bin` (в этом каталоге находится компилятор `javac.exe`). Между введенным путем и следующим не должно быть пробелов!

Теперь в области `Системные переменные` создайте переменную `CLASSPATH`. Для этого нажмите кнопку `Создать`. В открывшемся диалоговом окошке в поле `Имя переменной` введите `CLASSPATH`, а в поле `Значение переменной` задайте значение «`.;`» (точка и точка с запятой).



Подобным образом создается переменная `JAVA_HOME`. Значение этой переменной должно содержать путь к каталогу, в который была установлена Java, но без поддиректории `bin`. Например, значением `JAVA_HOME` может быть путь C:\Program Files\Java\jdk1.8.0_51.

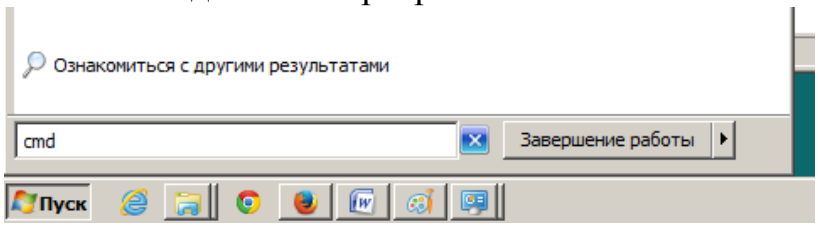
Изменения переменных вступают в силу после перезагрузки. После этого установку Java можно считать завершенной.

Теперь следует проверить корректность установки программной среды Java. Откройте командную строку. Ее можно запустить следующими способами:

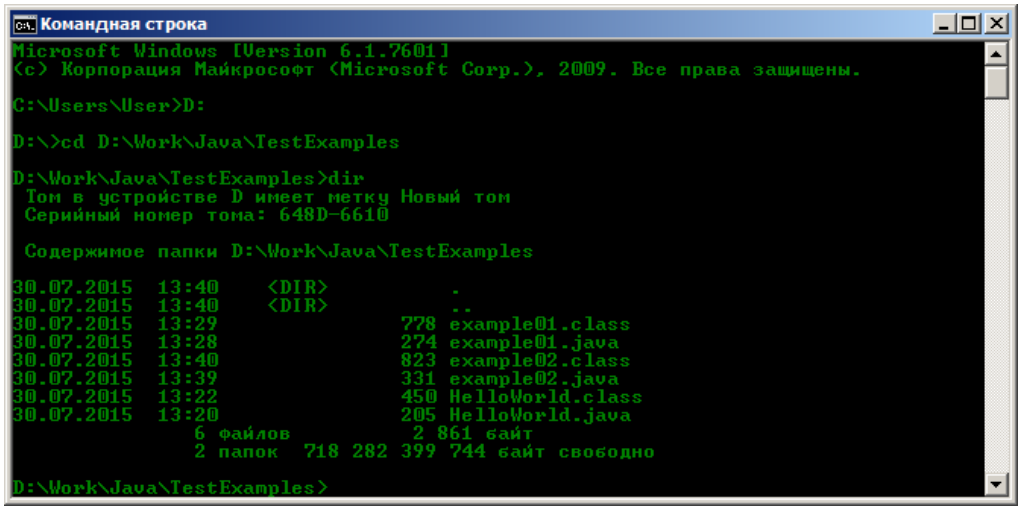
`Пуск->Все программы->Стандартные->Командная строка.`

или

`Пуск->Выполнить` и введите имя программы `cmd`.



Открывается окно консоли.



В этом окне в текстовом режиме подаются команды. Чтобы сменить текущий диск следует задать имя нового диска с двоеточием, например:

```
C:\Users\User>D:
```

Для того, чтобы перейти к нужному каталогу текущего диска, необходимо применить команду `cd` (**c**hange **d**irectory). Например, если вы желаете сделать активным каталог `TestExamples` с вашими примерами Java, то необходимо выполнить команду

```
D:>cd  D:\Work\Java\TestExamples
```

Имя текущего/активного каталога появится в символе приглашения, после которого будет мигать курсор ввода, например:

```
D:\Work\Java\TestExamples>_
```

Чтобы посмотреть содержимое активного каталога в окне консоли можно задать команду `dir`. Например,

```
D:\Work\Java\TestExamples>dir
```

Результат выполнения трех последних команд показан на предыдущем рисунке.

Теперь проверьте, что компилятор java (программа javac.exe) установлен корректно. Выполните команду

```
...>javac -version
```

Здесь три точки заменяют имя текущего каталога. После слова `javac` стоит пробел и минус. В ответ вы должны получить номер версии компилятора, например,

```
javac 1.8.0_51
```

Также проверяем установку виртуальной машины (интерпретатора байт кода)

```
...>java -version
```

В ответ вы должны получить три строчки текста.

Можно посмотреть значение переменных среды. Для этого надо выполнить команду `echo` и задать имя переменной в процентах. Например,

```
...>echo %JAVA_HOME%
C:\Program Files\Java\jdk1.8.0_51
```

Замечание. Иногда пути, прописанные в системных переменных, не должны содержать пробелов (это касается устаревших версий Java). Чтобы программная среда Java начала функционировать нормально пути, прописанные в системных

переменных, следует заключить в двойные кавычки. Тогда значение системной переменной Path может выглядеть, например, следующим образом (без единого пробела между путями):

"C:\Program Files\Java\ jdk1.8.0_51";C:\Program Files\MATLAB\R2010b\bin;

Теперь напишем и выполним вашу первую Java программу.

Пример 1. Создайте код файла HelloWorld.java в блокноте (или любом другом текстовом редакторе). Первые три строки – необязательные комментарии.

```
/**
* Самая первая программа. Файл назвать HelloWorld.java
*/
class HelloWorld {
    public static void main(String[] args)
    {
       System.out.println("Hello, World! It's my first program.");
    }
}
```

При сохранении имя файла и класса должны совпадать с точностью до регистра.

Чтобы откомпилировать программу, зайдите в консольном окне в каталог с этим файлом (используйте команду cd имя_каталога) и выполните команду

```
...>javac HelloWorld.java
```

(расширение имени файла обязательно).

Если ошибок нет, то файл HelloWorld.class создается в том же каталоге, где находится файл HelloWorld.java. Для запуска программы надо выполнить команду

```
...>java HelloWorld
```

(без расширения имени файла HelloWorld.class, а вместо троеточия будет стоять имя вашего каталога).

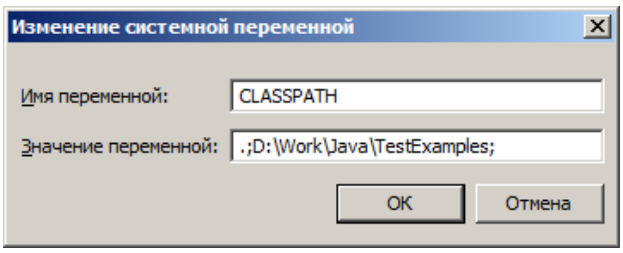
■

Замечание 1. Среда выполнения java ищет файлы в каталоге, указанном в переменной среды CLASSPATH. Установка значения «.» (точка) означает запуск файла из текущего каталога. Если при запуске интерпретатора java текущим будет не каталог, в котором размещен файл HelloWorld.class, то вы получите сообщение об ошибке. Для запуска программы следует сменить текущий каталог с помощью команды cd . Если вы не хотите менять текущий каталог, то программу можно запустить так

```
...> java –cp  полный_путь  имя_класса
```

(пробел после java, после –cp, после полного пути перед именем класса).

Если вы собираетесь создавать много программ в одном каталоге, то имя каталога можно задать вторым значением в переменной CLASSPATH. Например, измените ее значение следующим образом



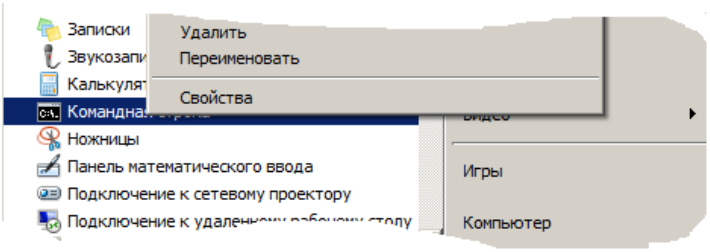
После перезагрузки вы сможете запускать ваши программы, расположенные в каталоге `D:\Work\Java\TestExamples`, даже тогда, когда текущим является другой каталог. Например,

```
C:\Users\User>java HelloWorld
```

Замечание 2. Многократные щелчки мышью для открытия консоли вам могут надоесть. Вызов командной строки можно ускорить, назначив ей комбинацию клавиш. Комбинация клавиш назначается в окне свойств этой команды. Для этого откройте

`Пуск->Все программы->Стандартные->Командная строка`

но не выполняйте команду. Сделайте по ней щелчок правой кнопкой мыши и выберите `Свойства`.



На закладке `Ярлык` поместите курсор в поле `Быстрый вызов` и нажмите комбинацию клавиш, например, **`Ctrl+Alt+W`**.

В следующем примере мы покажем, как отображать национальный шрифт.

Пример 2. Наберите следующий код в блокноте и сохраните файл под именем `example01.java`.

```
/**
* Файл назвать example01.java
*/
class example01 {
    public static void main(String[] args)
    {
     int x=10;
     int y=6;
     int z=x*y;
     System.out.println("Площадь прямоугольника со сторонами " +
               x + " см и " + y + " см равна " + z +" см2");
    }
}
```

Файл компилируем

```
…>javac example01.java
```

и запускаем на выполнение

```
…>java example01
```

После компиляции и выполнения все может выглядеть нормально. Но может случиться так, что в консольном окне вместо русского шрифта вы видите абракадабру. Для включения русского шрифта в консольном окне есть несколько способов. Следующий, самый предпочтительный и не требует менять код. Поэтому он подходит для любой программы. Запускаем консоль, щелкаем правой кнопкой мыши на заголовке окна и выбираем `Свойства`. На закладке `Шрифт` выбираем шрифт `Lucida Console` и жмем ОК. Далее набираем в консольном окне команду

```
...>ChCp 1251
```

В ответ получаем

```
Текущая кодовая страница: 1251.
```

Теперь запускаем Java программу и убеждаемся, что все работает.

```
…>java example01
```

Если ваша программа сразу нормально отобразила национальный шрифт, то описанных здесь «телодвижений» можно не делать.

■

Как мы говорили ранее, интерпретатор байт кода java ищет и запускает файлы из каталога, имя которого хранится в переменной среды `CLASSPATH`. Не всегда удобно иметь один `CLASSPATH` навсегда. Его можно изменить на время текущего сеанса работы консольного окна. Для этого в консольном окне можно выполнить команду, например,

```
...> set CLASSPATH=D:\Work\Java\Examples
```

Если в каталоге D:\ Work\Java\Examples создавать java-файлы, то компилятор javac будет в него помещать class-файлы. А интерпретатор java из него будет запускать class-файлы. Однако установка действует пока консольное окно открыто. При закрытии окна командной строки (консольного окна) установленные в сеансе переменные среды теряются (у нас это установки команды `ChCp 1251` и `set CLASSPATH=…`, если мы их устанавливали в текущем сеансе).

Есть еще один удобный способ работы с национальным шрифтом в консольном окне. Он доступен начиная с версии JDK 1.6. Для вывода в консоль можно использовать не стандартный поток вывода, а специальный класс `java.io.Console`. Этот класс всегда правильно определяет кодировку и, кроме того, обладает удобными методами чтения данных из консоли.

Пример 3. Вот измененный вариант 2-й программы.

```
/**
* Файл и класс назвать example03.java
*/
import java.io.*;

class example03 {
    public static void main(String[] args)
    {
     Console con=System.console();
     int x=10;
```

```
      int y=6;
      int z=x*y;
      con.printf("Площадь прямоугольника со сторонами " + x +
                          " см и " + y + " см равна " + z +" см2");
    }
}
```

Чтобы проверить работу шрифта откройте второе консольное окно (можно закрыть текущее и открыть новое). В нем не выполнялась команда изменения кодировки `ChCp 1251`. Если надо, выполните в этом окне команду

```
...>set CLASSPATH=D:\ Work\Java\TestExamples
```

Потом запустите программу

```
...>java example03
```

Русский текст должен нормально отображаться.

Пакетирование и распространение приложений Java.

Стандартный стиль программирования на java предполагает создание отдельного *.java файла для каждого класса. Соответственно, после компиляции для каждого класса будет создаваться свой *.class файл. Для запуска программы на другом компьютере надо будет перенести всю систему файлов приложения. Это не совсем удобно и, как правило, java программы распространяются в формате jar (а не *.exe) файла. Это обычный архивный файл, который создается утилитой jar.exe. Для собирания программы в один jar – файл ей (утилите jar) необходимо сообщить некоторую вспомогательную информацию. Для этого используется файл манифеста – обычный текстовый файл. Например, создадим файл HelloWorld.jar для программы из первого примера. В нашем простейшем случае файл манифеста будет содержать одну строку – имя стартового класса (в программе может быть много классов, но только один может содержать метод main). Вот содержимое файла манифеста для рассматриваемого примера

```
Main-Class: HelloWorld
```

Строчка должна заканчиваться символом новой строки (в конце нажмите Enter). Назовите этот файл, например, mainHelloWorld.txt. Для создания jar – архива из каталога, в котором находится файл HelloWorld.class, выполните команду

```
…>jar cfm HelloWorld.jar mainHelloWorld.txt HelloWorld.class
```

В текущем каталоге будет создан файл HelloWorld.jar. Для запуска программы теперь надо выполнить команду

```
…>java –jar HelloWorld.jar
```

Обычно в один jar файл помещают несколько файлов классов. Вот базовый формат команды для создания JAR-файла

```
…>jar cfm jar-файл файл_манифеста файл(ы)
```

Опция «c» показывает, что вы хотите *создать* (create) JAR-файл. Опция «m» показывает, что вы используете свой файл манифеста. Опция «f» показывает, что вы хотите направить вывод в *файл*, а не в стандартный поток вывода. Здесь `jar-файл` – это имя, которое вы хотите дать результирующему `jar-файлу`. Аргумент `файл(ы)` является списком с разделителем–пробелом из одного или

более имен файлов классов (*.class), которые вы хотите поместить в ваш JAR-файл. Аргумент `файл(ы)` может содержать также символ-джокер *. Если какие – то из входных–файлов являются каталогами, содержимое этих каталогов рекурсивно добавляется в архив JAR. В результате, команда сгенерирует сжатый JAR-файл и поместит его в текущий каталог. Таким образом, если архивный jar – файл должен содержать много файлов классов, то их имена надо перечислить через пробел в конце команды jar.

Опции `c`, `m` и `f` употребляются в любом порядке, но между ними не должно быть пробелов. Порядок букв `m` и `f` определяет порядок, в котором следуют имена файлов манифеста и архивного jar – файла.

Создание правильного файла манифеста для большого проекта является хлопотной задачей, ведь кроме написанных вами файлов классов в jar – файл необходимо включить также разные библиотеки, если они используются в вашем приложении. В следующем параграфе мы рассмотрим другой (автоматический) способ создания jar – файла.

Теперь скажем пару слов о текстовом редакторе. Блокнот (Notepad), которым вы пользовались для набора первых программ, является не лучшим решением. Есть много других неплохих текстовых редакторов. Рекомендуем обратить внимание на NotePad++. Он обеспечивает подсветку синтаксиса и работу с несколькими файлами. Выбор за вами.

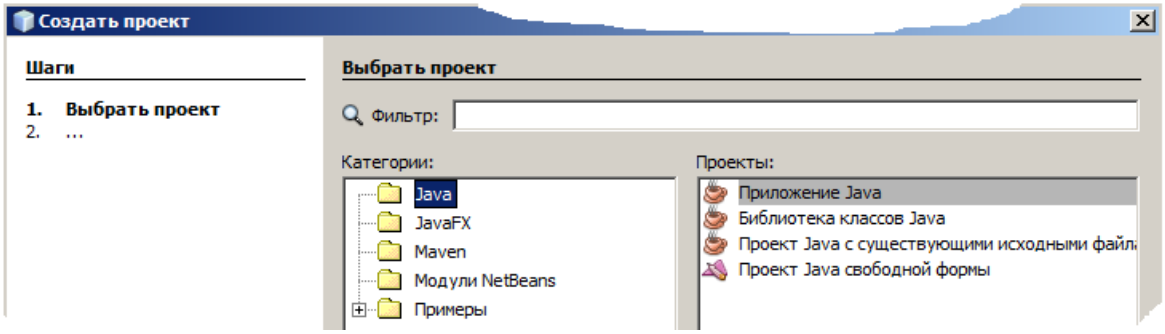
2. Интегрированная среда разработки NetBeans.

При разработке больших Java программ, состоящих из нескольких файлов и классов, использование компилятора командной строки не всегда удобно. Большинство разработчиков пользуется какой – либо интегрированной средой разработки. Одной из наиболее распространенных является бесплатная программа NetBeans IDE, которая поддерживается и спонсируется компанией Oracle, а разработка ведется независимым сообществом разработчиков.

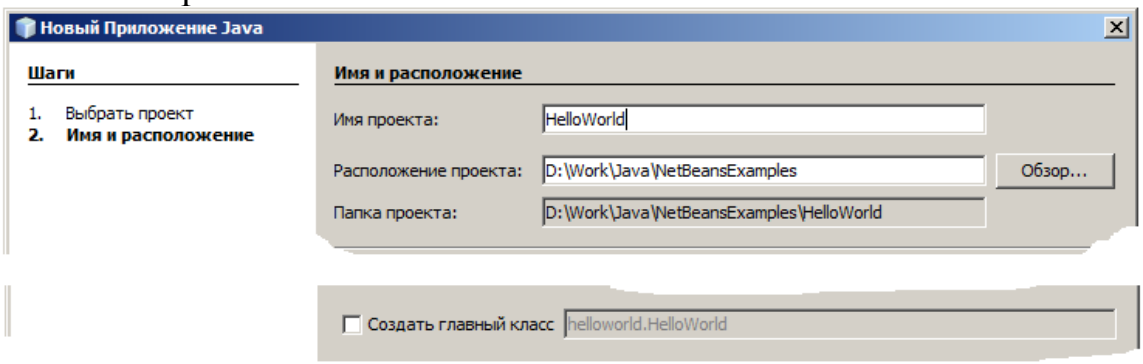
Мы полагаем, что среду NetBeans вы уже установили. Если нет, то лучше всего ее скачать с сайта https://netbeans.org/downloads. Выберите на этой странице язык IDE (например, русский), платформу (Windows), тип сборки интегрированной среды NetBeans (например, Java SE), и в соответствующем столбце нажмите кнопку `Загрузить`. Перед установкой среды NetBeans вы должны установить JDK. Запустите скачанный файл и следуйте инструкциям мастера установки.

Опишем ваши пошаговые действия при создания программы в NetBeans.

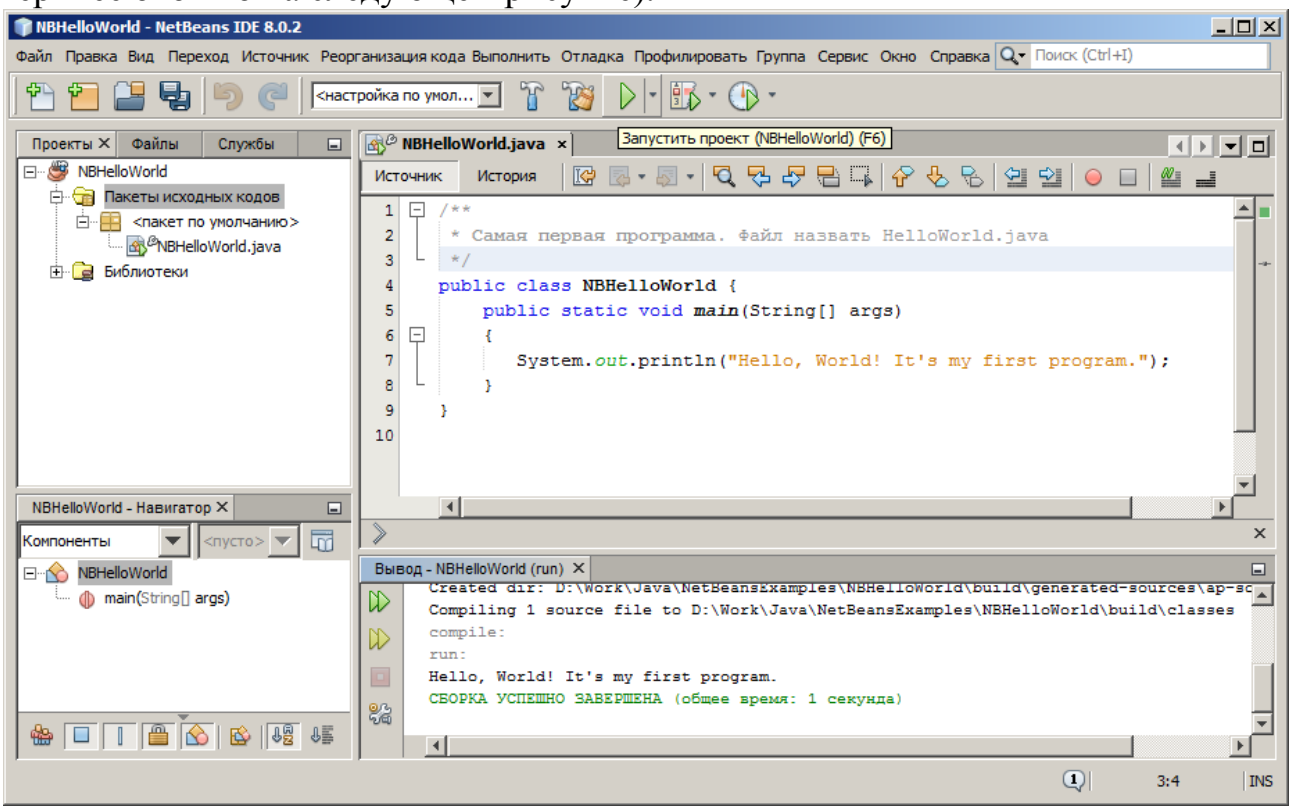
Пример 1. Запустите IDE NetBeans. Создайте новый проект. Для этого в меню `Файл` выберите `Создать проект...`. В открывшемся окне в разделе `Категории` выберите `Java`, а в разделе `Проекты` выберите `Приложение Java`.



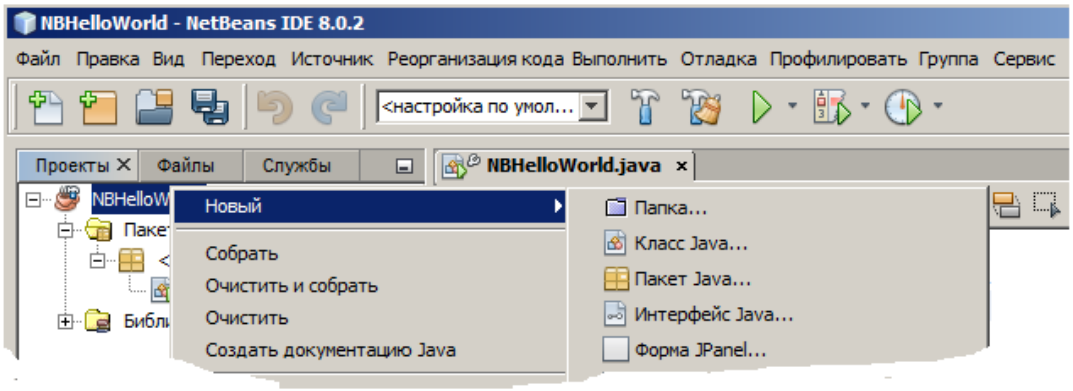
Нажмите кнопку `Далее`. На следующем шаге вы должны задать имя и расположение проекта.



Нажмите кнопку `Обзор` и выберите каталог, в котором вы будете сохранять ваши проекты. В поле `Имя проекта` введите HelloWorld. Снимите также галочку `Создать главный класс` (или не снимайте, а введите в соответствующем поле имя HelloWorld – в этом случае за вас немного потрудится IDE среда). Нажмите кнопку `Готово`. Проект откроется в среде IDE и его логическая структура будет отображена в панели `Проекты` (левое верхнее окошко на следующем рисунке).



В окне Проекты щелкните правой кнопкой мыши по имени проекта HelloWorld, выберите Новый – Класс Java... и, в открывшемся окне, введите имя класса HelloWorld.



Справа в IDE откроется редактор кода с файлом HelloWorld.java. Если в нем есть какой либо текст, то измените его так, чтобы он соответствовал коду примера 1 предыдущего параграфа.

```
/**
* Самая первая программа. Файл назвать HelloWorld.java
*/
class HelloWorld {
    public static void main(String[] args)
    {
       System.out.println("Hello, World! It's my first program.");
    }
}
```

Если на втором шаге вы не снимали флажок Создать главный класс, то окно редактора сразу было бы создано с кодом пустого класса HelloWorld.

Теперь отключите компиляцию при сохранении проекта. Для этого щелкните проект правой кнопкой мыши и выберите Свойства. В окне Свойства перейдите на вкладку Компиляция и снимите флажок Компиляция при сохранении. Закройте окно Свойства кнопкой OK.

Сохраните ваш проект путем выбора команды Файл – Сохранить.

Для запуска программы выберите Выполнить – Запустить проект. Если ошибок в программе нет, то в правом нижнем окне Вывод – HelloWorld(run) вы увидите сообщения, подобные приведенным на предпоследнем рисунке.

Поздравляем! Вы создали и выполнили вашу первую программу в IDE NetBeans. Если вы войдете в каталог проекта HelloWorld, созданного средой NetBeans, то найдете в нем подкаталог ...\build\classes, который будет содержать откомпилированный файл HelloWorld.class. Попробуйте запустить его не из интегрированной среды, а с помощью интерпретатора java так, как мы это делали в предыдущем параграфе.

Пример 2. Запустите IDE NetBeans. Создайте новый проект с именем Welcome. Выполните последовательность действий такую же, как в предыдущем примере, но на втором шаге, когда вы задаете имя проекта и его расположение,

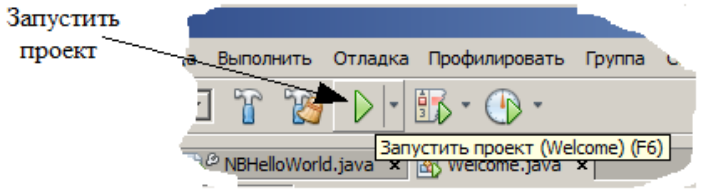
не снимайте флажок Создать главный класс, а введите в соответствующем поле имя класса Welcome. Обратите внимание, что в панели Проекты логическая структура предыдущего проекта остается, так же как в окне редактора кода остается закладка с кодом файла HelloWorld.java. Файлы кода в редакторе вы можете закрыть «крестиком» на соответствующих закладках, а старые проекты можно удалить из окна проектов, если щелкнуть правой кнопкой мыши по имени проекта и в выпадающем списке выбрать команду Закрыть.

В окне редактора измените код файла Welcome.java так, как показано ниже

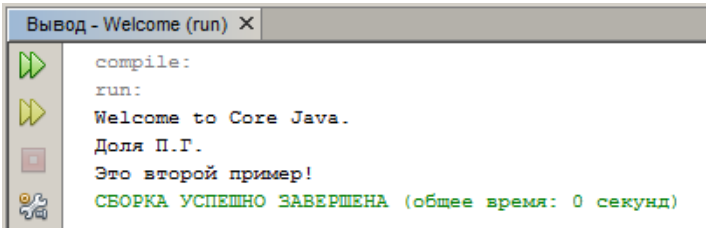
```
// файл Welcome.java
public class Welcome {
    public static void main(String[] args) {
        String[] greeting=new String[3];
          greeting[0]="Welcome to Core Java.";
          greeting[1]="Доля П.Г.";
          greeting[2]="Это второй пример!";

          for (int i=0;i<3; i++)
               System.out.println(greeting[i]);
    }
}
```

Мы здесь не поясняем синтаксис языка Java. Надеемся, что в приведенном коде все понятно. Если нет, то вам следует обратиться к любому учебнику по языку Java.

Для запуска программы выберите Выполнить – Запустить проект или на панели инструментов нажмите соответствующую кнопку. При этом, в окне Проекты должен быть выделен тот проект (или один из его элементов), с которым вы работаете.



Если ошибок в программе нет, то в панели Вывод вы увидите сообщения, подобные приведенным ниже.



В каталоге проекта, созданного средой NetBeans, в подкаталоге ...\build\classes будет создан файл Welcome.class. Запустите его не из интегрированной среды, а с помощью интерпретатора java и убедитесь, что национальный шрифт в окне консоли отображается верно (а не только в окне вывода IDE среды).

■

Теперь напишем программу, состоящую из нескольких классов. В нашем демонстрационном примере будет два очень примитивных класса. Однако в реальных приложениях классов бывает много и они содержат большое количество методов.

Пример 3. Запустите IDE NetBeans. Создайте новый проект с именем CompositeApp. Выполните последовательность действий такую же, как в предыдущем примере. На втором шаге не снимайте флажок `Создать главный класс` и введите имя CompositeApp. Если желаете, то в окне `Свойства` проекта можете отключить компиляцию при сохранении.

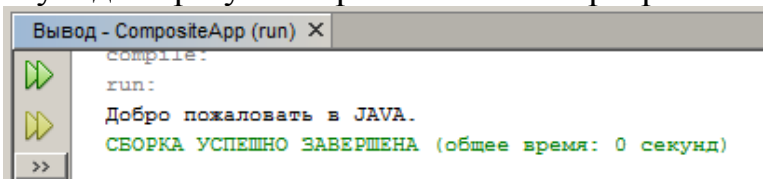
В нашей программе будет главный класс CompositeApp. Для него файл CompositeApp.java уже создан средой разработки, когда на втором шаге вы оставили флажок `Создать главный класс`. Вспомогательный класс будет содержать текстовую строку приветствия и метод для ее отображения. Для создания второго класса поступите так, как вы делали в первом примере. В окне `Проекты` щелкните правой кнопкой мыши по имени проекта CompositeApp, выберите `Новый – Класс Java...` и, в открывшемся окне, введите имя класса Output. В окне редактора будет создан файл Output.java. Измените его так, чтобы он содержал следующий код

```
// Класс для вывода информации. Файл Output.java
 public class Output {
    String greeting="Добро пожаловать в JAVA.";
    public void dataoutput() {
        System.out.println(greeting);
    }
}
```

Теперь откройте файл CompositeApp.java и измените его так, чтобы он соответствовал следующему тексту

```
// Приложение из нескольких классов. Файл CompositeApp.java
public class CompositeApp {
    public static void main(String[] args) {
        Output out=new Output();
        out.dataoutput();
    }
}
```

Сохраните проект и запустите его на выполнение. Если ошибок не было, то в панели вывода вы увидите результат работы вашей программы



В каталоге проекта, созданного средой NetBeans, в подкаталоге ...\build\classes будет создано два файла: CompositeApp.class и Output.class. Зайдите в консольном окне в этот каталог и запустите приложение командой

```
...>java CompositeApp
```

Все нормально работает. Но, если вам потребуется передать программу пользователю, то придется копировать весь каталог с файлами ваших классов.

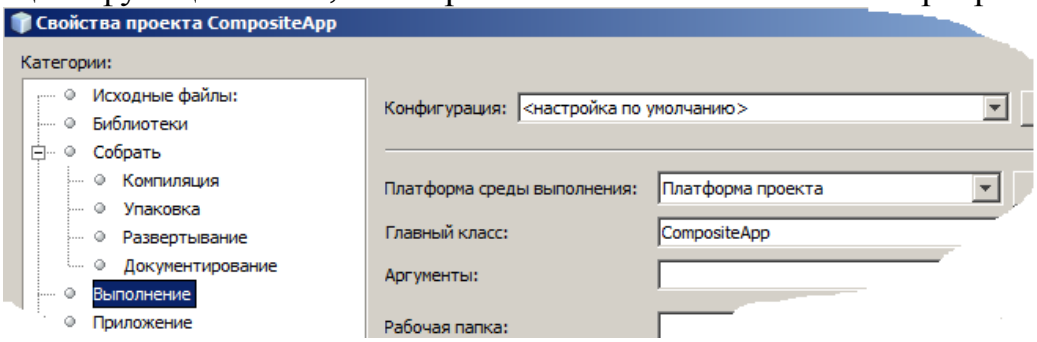
В предыдущем параграфе мы уже говорили, что избежать этого помогает jar – архив, который можно запускать не распаковывая. Он создается с помощью утилиты jar, но для больших проектов ее использование довольно хлопотно и интегрированная среда разработки NetBeans может облегчить вам жизнь. Для создания jar – файла надо выбрать команду `Выполнить – Собрать проект` или в качестве альтернативы в панели `Проекты` щелкнуть правой кнопкой мыши узел проекта и выбрать команду `Собрать`. В результате в каталоге проекта появится подкаталог ...\dist, в котором будет создан файл CompositeApp.jar.

Работоспособность программы можно проверить из командной строки

```
...>java –jar CompositeApp.jar
```

Это значит, что ваше приложение может работать уже без интегрированной среды и все, что нужно для его распространения – это копирование файла CompositeApp.jar на компьютеры пользователей.

При сборке проекта автоматически создается файл манифеста. Его содержимое можно просмотреть в панели `Файлы`. Для этого надо перейти к узлу dist\CompositeApp.jar этого окна. Затем разверните узел jar – файла, в котором разверните папку META-INF и дважды щелкните MANIFEST.MF, чтобы отобразить содержимое манифеста в редакторе кода. Обратите внимание, что созданный в результате сборки файл манифеста «знает», какой класс является стартовым. Чтобы его установить (или проверить установку) откройте окно `Свойства` проекта и выберите категорию `Выполнение`, в которой есть поле `Главный класс`. В этом поле вы можете задать имя класса, содержащего функцию main, с которой начинается выполнение программы.



В нашем случае установка этого класса была выполнена на втором шаге создания проекта, когда вы задали имя главного класса.

Если вы внесли в проект изменения, то jar – файл можно обновить командой `Выполнить – Очистить и собрать проект` или командой `Очистить и собрать` в выпадающем списке команд проекта, который открывается при щелчке по его узлу правой кнопкой мыши. По этой команде удаляются предварительно скомпилированные файлы и другие результаты сборки, а затем выполняется повторная компиляция приложения и формируется новый jar – файл.

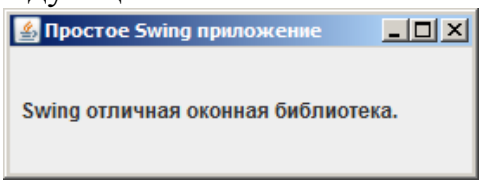
■

Теперь создадим простое оконное приложение java.

Пример 4. Процесс создание оконных приложений java отличается от создания консольных приложений только использованием других библиотечных функций. Запустите IDE NetBeans. Создайте новый проект с именем swing01. На втором шаге оставьте флажок `Создать главный класс` и введите имя главного класса Swing01. В окне редактора будет создан файл Swing01.java. Отредактируйте его так, чтобы он содержал следующий код

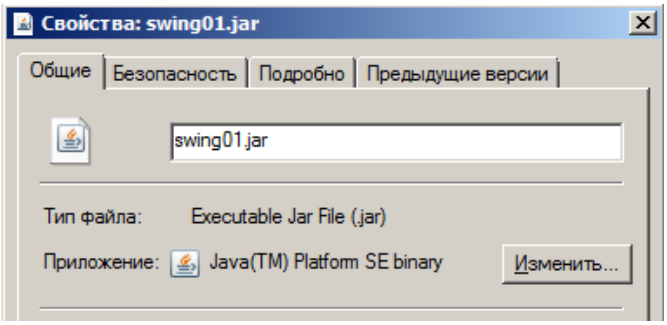
```
// файл swing1.java
import javax.swing.*;

public class Swing01 {
  Swing01() {                          //Конструктор класса
    // Создать новый JFrame контейнер (окно)
    JFrame jfrm = new JFrame("Простое Swing приложение");
    jfrm.setSize(275, 100);  // Задать начальные размеры окна
    //Завершать программу, когда пользователь закрывает приложение
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    // Создать текстовую метку.
    JLabel jlab = new
              JLabel("  Swing отличная оконная библиотека.");
    jfrm.add(jlab);                    // Добавить метку в окно.
    jfrm.setVisible(true);             // Отобразить окно.
  }

    public static void main(String[] args) {
      // Создать окно в потоке обрабатывающем сообщения.
      SwingUtilities.invokeLater(new Runnable()
                {public void run(){new Swing01();}});
    }
}
```

В нашем классе `Swing01` содержится два метода: `main` и конструктор класса `Swing01`. В конструкторе создается окно приложения с помощью объекта класса `JFrame`. Окну задается заголовок, размер и способ реакции на событие закрытия окна. Также внутрь окна помещается простая текстовая метка. В методе `main` создается объект нашего класса `Swing01`. Если в приведенном коде не все понятно, то вам следует обратиться к любому учебнику по языку Java, в котором есть описание библиотеки Swing. В результате работы программы вы увидите следующее окно



Теперь создадим jar – файл. Для этого в панели `Проекты` щелкните правой кнопкой мыши узел проекта и выберите команду `Собрать`. В результате в каталоге проекта появится подкаталог ...\dist, в котором будет создан файл swing01.jar. Закройте IDE и в Windows откройте эту папку. Сделайте двойной

щелчок мышью по файлу swing01.jar. Он должен запуститься. Если этого не произошло, то щелкните по файлу правой кнопкой мыши, в выпадающем списке выберите Свойства и на закладке Общие нажмите кнопку Изменить...



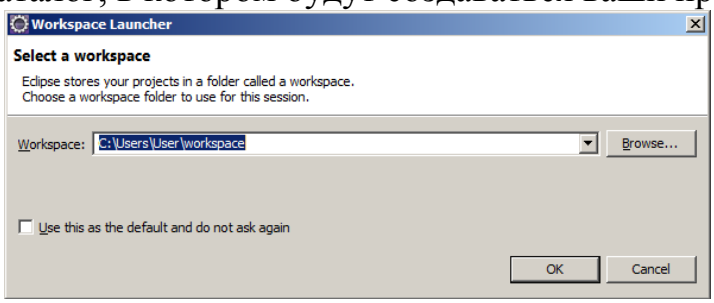
В открывшемся окне выберите программу «Java(TM) Platform SE binary» для открытия этого файла и установите флажок «Использовать выбранную программу для всех файлов такого типа». Теперь все jar – файлы будут запускаться двойным щелчком мыши.

3. Интегрированная среда разработки Eclipse.

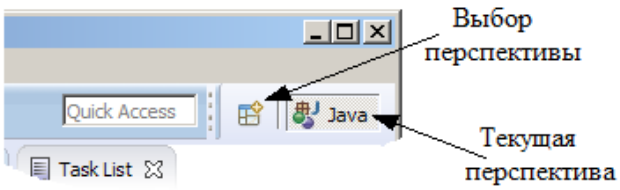
В качестве альтернативы мы можем предложить вам использовать другую бесплатную интегрированную среду разработки java программ – IDE Eclipse. Первоначально среда Eclipse была создана фирмой IBM, но потом была передана для развития независимому сообществу.

Мы полагаем, что Eclipse вы уже установили. Если нет, то лучше всего ее скачать с сайта https://eclipse.org/downloads. При этом никакой установки не требуется, просто распакуйте скачанный архив в какую – либо папку, например, D:\Eclipse. Найдите в ней файл eclipse.exe и создайте для него ярлык на рабочем столе.

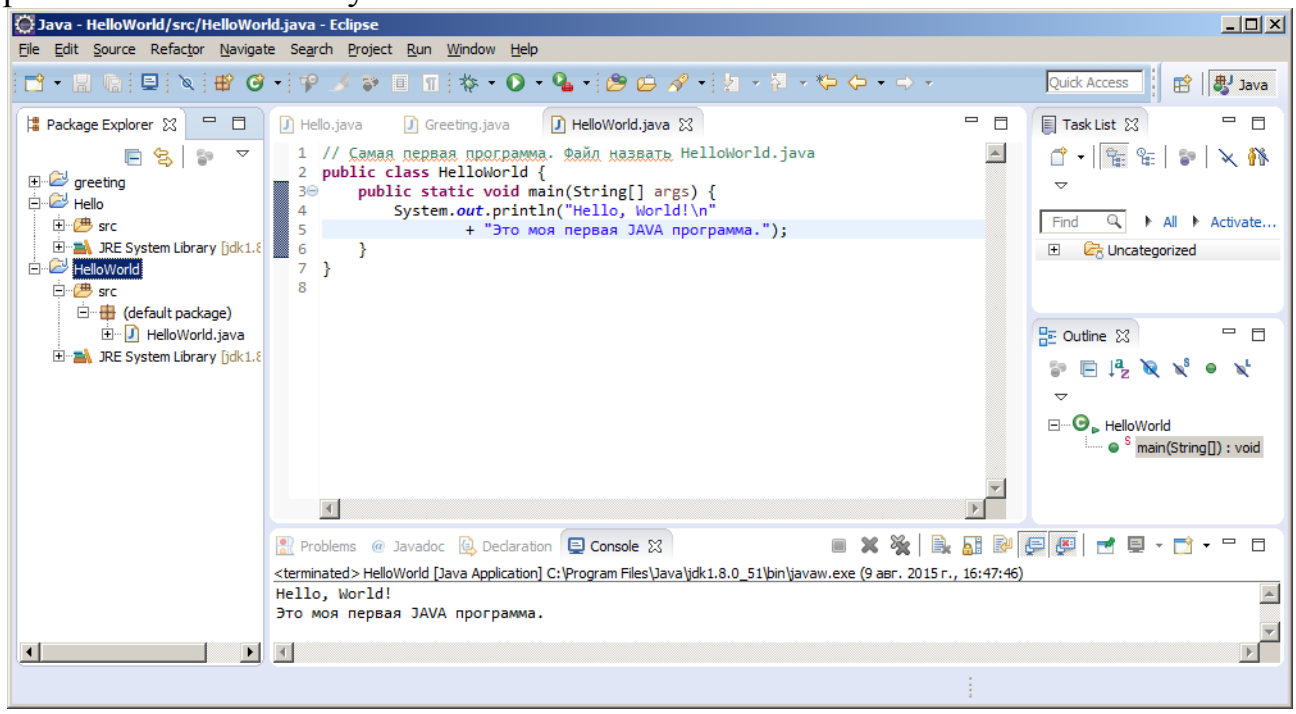
Запустите программу. При первом запуске вам будет предложено выбрать Workspace – это каталог, в котором будут создаваться ваши проекты.



Нажмите кнопку Browse и выберите каталог. При первом запуске сразу после этого будет открыто окно приветствия. Для перехода к интерфейсу щелкните по ссылке Workbench в правом верхнем углу. Откроется основное окно программы. В его правом верхнем углу щелкните по кнопке Open Perspective и в открывшемся окне выберите перспективу Java.



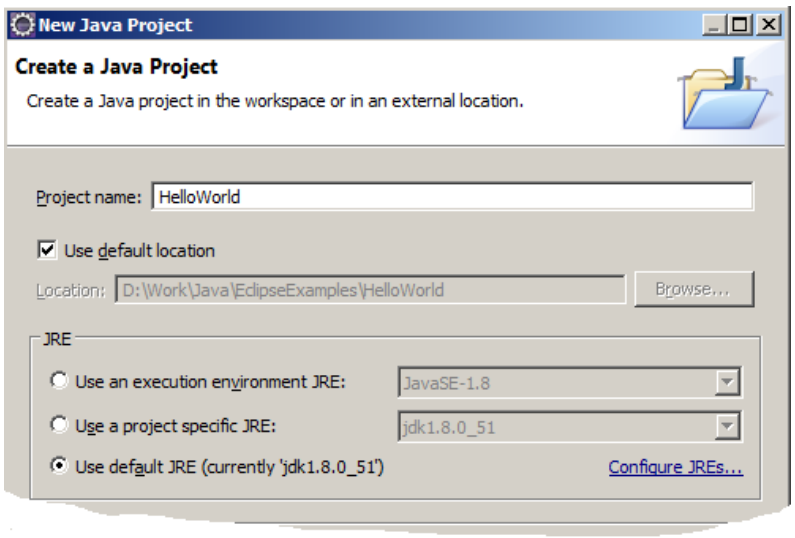
Perspectives (перспективы) определяют набор и расположение окон и панелей инструментов интегрированной среды. Позже вы сможете сами управлять ими. В результате главное окно программы может изменить свой вид. Теперь оно будет состоять из строки меню и панели инструментов, расположенных вверху, окна проекта (Package Explorer) слева, окна редактора в центре, панелей Task List и Outline, расположенных справа одна под другой, и панели вывода, расположенного внизу.



Перейдем к созданию проекта.

Пример 1. Программа HelloWorld.

Зайдите в меню `File – New – Java Project`. В открывшемся окне New Java Project введите имя проекта HelloWorld и убедитесь, что в группе JRE выбрана опция Use default JRE.



Остальные установки не меняйте и нажмите кнопку Next. В следующем окне ничего не меняйте и жмите кнопку Finish. В панели Package Explorer появится логическая структура пустого проекта. Теперь создадим класс. Для этого щелкаем правой кнопкой мыши на узле проекта `src` и выбираем `New->Class`. Можно также на панели инструментов использовать кнопку `New Java Class`. В открывшемся окне в поле Name вводим имя класса, например, HelloWorld и для создания метода main устанавливаем флажок public static void main(String[] args). Другие флажки снимаем и жмем кнопку Finish. В окне редактора будет создан файл HelloWorld.java с заготовкой кода главного класса. Измените его так, чтобы он соответствовал коду примера 1 первого параграфа.

```
// Первая программа. Файл назвать HelloWorld.java
public class HelloWorld {
     public static void main(String[] args) {
          System.out.println("Hello, World!\n"
                    + "Это моя первая JAVA программа.");
     }
}
```

Сохраните ваш проект путем выбора команды меню `File – Save All`.

Для запуска программы выберите команду меню `Run – Run`. Откроется окно Save and Launch, которое переспросит имя проекта (в IDE может быть открыто несколько проектов). Жмите кнопку OK и, если ошибок в программе нет, то в панели вывода (внизу главного окна IDE) вы увидите сообщения, подобные приведенным на предпоследнем рисунке.

Если вы войдете в каталог проекта HelloWorld, созданного средой Eclipse, то найдете в нем подкаталог ...\bin, который будет содержать откомпилированный файл HelloWorld.class. Попробуйте запустить его не из интегрированной среды, а с помощью интерпретатора java так, как мы это делали в первом параграфе.

Пример 2. Программа запрашивает имя и выводит приветствие.

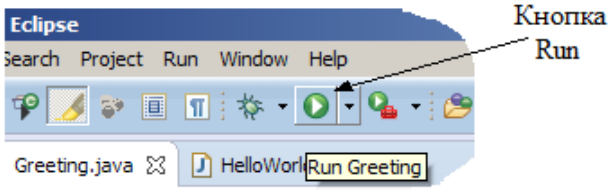
Запустите IDE Eclipse. Создайте новый проект с именем greeting. Выполните последовательность действий такую же, как в предыдущем

примере, и создайте главный класс с именем Greeting. Обратите внимание, что в панели Package Explorer логическая структура предыдущего проекта остается, так же как в окне редактора кода остается закладка с кодом файла HelloWorld.java. Файлы кода в редакторе вы можете закрыть «крестиком» на соответствующих закладках, а старые проекты можно удалить из панели проектов, если щелкнуть правой кнопкой мыши по имени проекта и в выпадающем списке выбрать команду `Delete`.

В окне редактора измените код файла Greeting.java так, как показано ниже

```
// файл Greeting.java
import java.io.*;

public class Greeting {
     static int b;
     static String sname;
     static char[] charray;

     public static void main(String[] args) {
          charray=new char[20];
          int i=0;
          System.out.println("What is Your name?");
          try {
               while((b=System.in.read())!=13)
               {
               charray[i++]=(char) b;
               }
          }
          catch(IOException e) {
               System.out.println("The error "+e);
          }
          sname=new String(charray);
          System.out.println("Hello, " + sname);
     }
}
```

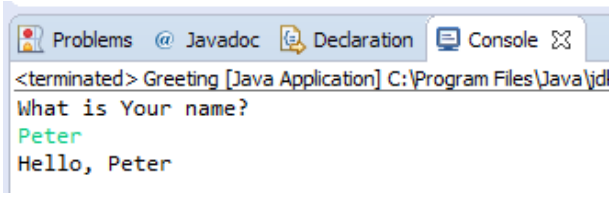
Сохраните проект и запустите его на выполнение командой меню `Run – Run` или на панели инструментов нажмите соответствующую кнопку.



Будьте внимательны. Кнопка `Run` может запустить предыдущий проект. Чтобы вы не сомневались, какой проект будет компилироваться и выполняться, первый раз запуск проекта выполните из выпадающего списка команд проекта. Для этого щелкните правой кнопкой мыши по проекту в панели Package Explorer и выберите команду `Run As – Java Application`. Если ошибок в программе нет, то в окне вывода вы увидите приглашение

```
What is Your name?
```

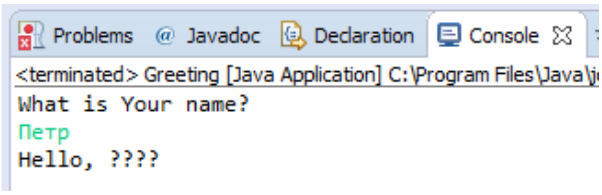
и приложение будет ожидать вашего ответа. Например, введите `Peter` и нажмите Enter. В результате в панели вывода вы увидите сообщения, подобные приведенным ниже.



В каталоге проекта, созданного средой Eclipse, в подкаталоге ...\bin будет создан файл Greeting.class. Зайдите в консольном окне в этот каталог и запустите приложение командой

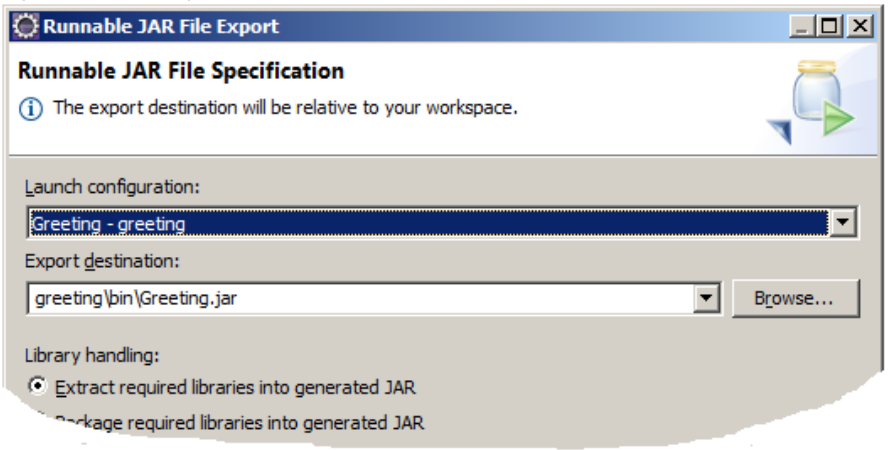
```
...>java Greeting
```

У приведенной программы есть один недостаток. Если в ответ на приглашение вы введете имя на русском языке, то увидите сообщения, подобные приведенным ниже.



Наша программа некорректно работает с национальными шрифтами! Этот недостаток связан с тем, что консоль понимает только однобайтовую кодировку ср866, а функции java работают с двухбайтовыми кодами символов. Есть несколько способов преодолеть этот недостаток. Мы не будем на этом останавливаться, поскольку в оконных приложениях этой проблемы нет.

Теперь создадим архивный jar – файл. Для этого надо щелкнуть правой кнопкой мыши узел проекта и выбрать команду `Export`. В открывшемся окне выбираем `Java->Runnable JAR File` и жмём кнопку Next. В открывшемся окне в выпадающем списке Launch configuration выбираем `Greeting – greeting`. В поле Export destination следует указать имя jar – архива и папку, где он будет находиться.



Остальные настройки оставляем без изменения и жмем кнопку Finish. Теперь наш архив можно запускать командой

```
...>java -jar Greeting.jar
```

Пример 3. Двумерная графика в Windows программе.

Запустите IDE Eclipse. Создайте новый проект с именем shapes и главный класс с именем Shapes. В окне редактора измените код файла Shapes.java так, как показано ниже

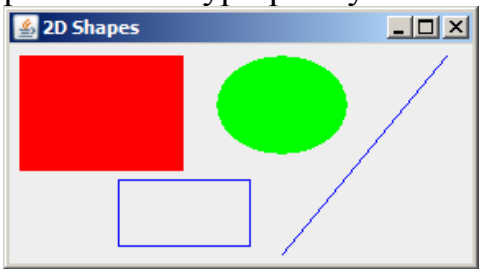
```
// файл Shapes.java
import java.awt.*;
import javax.swing.*;

public class Shapes extends JFrame  {
     public Shapes()
     {
          super("2D Shapes");
     }

     public void paint(Graphics g) {
          super.paint(g);
          g.setColor(new Color(255,0,0));
          g.fillRect(10,30,100,70); // красный прямоугольник
          g.setColor(new Color(0,255,0));
          g.fillOval(130,30,80,60);     // зеленый эллипс
          g.setColor(new Color(0,0,255));
          g.drawLine(170,150,270,30);   // синий отрезок
          g.drawRect(70,105,80,40); // синий контур прямоугольника
     }

     public static void main(String[] args) {
          Shapes application=new Shapes();
          application.setSize(290, 160);
          application.setDefaultCloseOperation(
                                        JFrame.EXIT_ON_CLOSE);
          application.setVisible(true);
     }
}
```

Сохраните проект и запустите его на выполнение.

В каталоге проекта, созданного средой Eclipse, в подкаталоге ...\bin будет создан файл Shapes.class. Зайдите в консольном окне в этот каталог и запустите приложение командой

```
...>java Shapes
```

Откроется окно программы, в котором нарисованы красный прямоугольник, зеленый эллипс, синие отрезок и контур прямоугольника



Участки кода, рисующие эти фигуры, в файле сопровождаются соответствующими комментариями.

В методе main вначале создается объект класса Shapes. Этот класс наследуется из класса `JFrame`, который определяет прямоугольное окно, содержащее строку заголовка, кнопки закрытия, сворачивания и разворачивания окна, а также системное меню. Далее задаются размеры окна в пикселях.

Потом задается реакция окна на его закрытие. По умолчанию, когда окно верхнего уровня закрывается (например, по щелчку на кнопке с крестиком в верхнем правом углу), оно удаляется с экрана, но приложение не завершается. В большинстве случаев такое поведение неприемлемо. Обычно желательно, чтобы при закрытии окна верхнего уровня приложение завершалось. Вызов метода setDefaultCloseOperation( ) с аргументом – константой JFrame.EXIT_ON_CLOSE определяет именно такое поведение.

Четвертая команда в методе main обеспечивает отображение окна на экране. По умолчанию окно не видимо, поэтому для его отображения требуется вызов метода setVisible(true).

Рисование фигур мы выполняем в методе paint класса Shapes. Этот метод перегружает (заменяет) метод paint базового класса JFrame. Он вызывается системой каждый раз, когда окно появляется первый раз, меняет свои размеры или повторно открывается. У этого метода единственным аргументом является объект класса Graphics, который имеет ряд методов рисования. Некоторые из них использованы в нашей примитивной программе.

Теперь создайте jar – архив так, как мы это делали в предыдущем примере. После этого программу можно запускать двойным щелчком мыши по имени jar – файла.

■

Вот несколько полезных комбинаций клавиш редактора кода:

- Ctrl + / — позволяет закомментировать выделенную часть кода;
- Ctrl + Пробел — автозаполнение;
- Ctrl+ Shift + F — автоформатирование, убирает беспорядок в выделенной части кода.

Заключение.

В нашем небольшом пособии мы дали введение в способы создания java приложений. Вы написали несколько простых программ и поняли, как их можно переносить на компьютеры пользователей. Однако, это только первый шаг длинnouго пути. Желаем удачи!