

Модуль 1. Основы программирования

Тема 1.1. Здравствуй Мир!

2 часа

Оглавление

[О Курсе IT ШКОЛА SAMSUNG](#)

[Обозначения](#)

[1.1. Здравствуй, мир!](#)

[1.1.1. Среда программирования](#)

[1.1.2. Первая программа](#)

[1.1.3. Запуск на мобильном устройстве](#)

[Задание 1.1.1](#)

[Благодарности](#)

О Курсе IT ШКОЛА SAMSUNG

Современное программирование нельзя изучать последовательно.

В наше время электронно-вычислительные устройства сильно уменьшаются в размерах. Практически все изучающие этот курс, носят компьютер, (возможно, и не один!) в кармане, ведь мобильные телефоны сейчас - это самые настоящие компьютеры. При этом их возможности, наоборот, возрастают очень быстрыми темпами.

Вычислительная мощность современного мобильного телефона серьезно превосходит общую вычислительную мощность компьютеров институтов, в которых рассчитывали первые космические полеты. Это дает возможность использовать их в том качестве, о котором раньше и не думали. Электронные устройства, в том числе мобильные, давно перестали быть только «вычислительными». Удивительно, что одно устройство весом значительно меньше килограмма является и видеоплеером и игровой приставкой и навигатором и, конечно, средством беспроводной коммуникации!

Мир программирования, магическая составляющей компьютерных технологий, еще более удивителен и разнообразен. Как может быть часть сложнее целого? Просто магия не задумывается, как оно там устроено внутри, а мы в IT-Хогвартсе будем говорить именно об этом. Обычно на старте хочется «всего и сразу». Больше того, это самое «все и сразу» здесь просто необходимо. Невозможно ребенку сначала научиться ходить, а потом есть ложкой, нужно это делать одновременно. Поэтому в тексте будут иногда встречаться незнакомые понятия. Скорее всего, они будут объяснены позже. Впрочем, часто незнакомое интуитивно понятно. Кроме того, разработчикам под Android непременно нужно уметь пользоваться Google. А еще нужно запастись терпением и быть готовым к тому, что иногда в курсе будут скучные моменты. Но ведь невозможно быть лучшим в профессии не зная всех деталей?

Обозначения



В таких блоках выделяются дополнительные сведения, необходимые для глубокого понимания материала.

Может быть, при первом чтении их стоит пропускать, но затем обязательно возвращаться и пытаться осмыслить.



В таких блоках показываются возможности среды программирования Android Studio, которую мы будем преимущественно использовать при программировании Android-приложений.

Эти сведения можно использовать и для работы в IntelliJ IDEA, эти две среды очень похожи по интерфейсу.



В таких блоках показываются возможности среды программирования Eclipse, которую мы будем использовать на протяжении курса.

Вам, конечно, известно, что Google в качестве основной IDE для разработки под Android использует Android Studio, а Eclipse больше не поддерживает.

Почему же мы используем Eclipse? Android Studio на первых занятиях не используется по двум причинам.

Во-первых, на ней нельзя написать “обычное” простое консольное приложение, а первый модуль курса - изучение основ Java. Можно, конечно, все в TestBed запускать, но это только расход времени.*

Еще одна причина - Eclipse долгое время был фактически стандартной средой разработки на Java, в том числе под Android. Много статей и руководств ориентированы на него. Поэтому иметь представление об этой среде, навыки работы в ней, очень полезно.

На компьютерах Школы и в дистрибутивах, которые предлагаются учащимся Samsung IT School для установки на домашние компьютеры есть три среды - Eclipse, Android Studio и IntelliJ IDEA, из которой и “выросла” Android Studio, поэтому они похожи по интерфейсу. Но IntelliJ IDEA работает несколько медленней Eclipse.

****TestBed** - Android-проект, разработанный инженерами Samsung совместно с преподавателями Школы, позволяющий запускать консольные программы на экранах мобильных устройств.*

1.1. Здравствуй, мир!

Для того, чтобы увидеть результат программы, пусть даже самой простой нужно:

1. **Написать программу.** Для этого нужно владеть языком программирования, причем знать не только синтаксис, но и возможности так называемых библиотек, уметь пользоваться текстовым редактором, при необходимости установить его, сохранить файл на диске
2. **Скомпилировать.** Процессор понимает только машинные команды, поэтому нужно перевести текст программы в числовые коды. Это не всегда происходит сразу после написания кода. Во многих языках программирования окончательная компиляция происходит уже в процессе выполнения программы.
3. **В случае, если программа разрабатывается для другого устройства, загрузить откомпилированный (или частично откомпилированный код) на устройство.** Для этого нужно связаться с устройством, использовать протоколы передачи информации и
4. **Запустить программу на выполнение.** Для того, чтобы на хорошем уровне понимать детали этого процесса нужно учиться не один год. И тем не менее, мы напишем, скомпилируем, загрузим и запустим первую программу уже на этом занятии. Как это возможно? Существует много средств для облегчения труда разработчиков. При этом нужно постоянно осознавать, что нужно понимать, как «оно устроено внутри». Мы будем возвращаться и рассматривать детали постепенно.

1.1.1. Среда программирования

Прежде всего, главный инструмент разработчика это **среда программирования** (еще называют “среда разработки”). Для эффективной работы необходимо хорошо представлять ее возможности и постоянно пользоваться ими.

Обычно среда разработки содержит несколько компонент.

Редактор кода. Главное преимущество редактора среды программирования перед обычным редактором, например, Блокнотом в Windows – в подсветке синтаксиса – ключевые слова выделяются цветом.

Компилятор. Среда может ставиться и без компилятора и работать с различными компиляторами, установленными на компьютере, но в любом случае она делает процесс компиляции «прозрачным» для программиста. Например, среда *Eclipse*, по умолчанию настроена так, что компиляция и запуск программ объединены в одной команде (делаются одной кнопкой).

Средства запуска. *Eclipse* с установленным плагином ADT (Android Developers Tools) позволяет запускать программы сразу на устройстве или на эмуляторе.

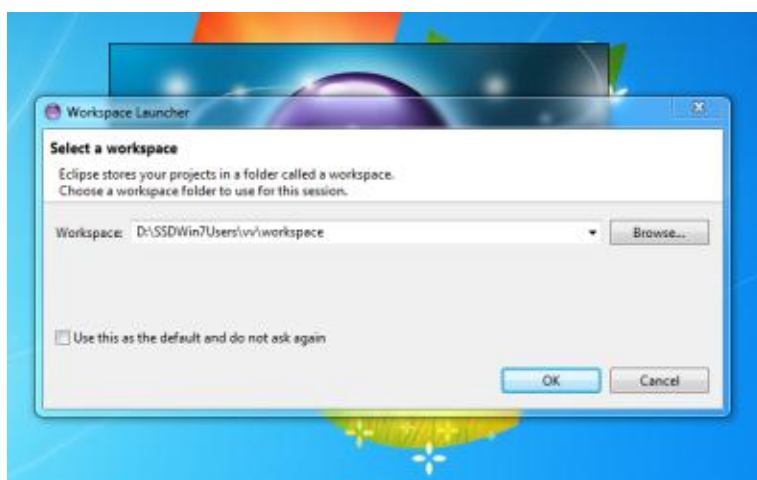
Отладчик. Для того, чтобы разобраться в том, почему же программа работает не так, как задумывал разработчик, можно не только пристально смотреть в код, но и выполнить программу по шагам, чтобы посмотреть промежуточные результаты работы и понять, правильно ли ведет себя программа. Все это делает отладчик. Без отладчика (как, впрочем и без среды программирования) в принципе можно обойтись, но грамотное его использование серьезно облегчает работу.

Для того чтобы установить компилятор *Java*, среду *Eclipse* и средства разработчика под *Android* можно воспользоваться сборкой школы Samsung (см. в курсе системы обучения). После запуска инсталлятора необходимо ответить на несколько стандартных вопросов и можно начинать разработку. Когда-нибудь мы обязательно подробно рассмотрим состав установленного пакета и попробуем установить нужные компоненты по отдельности самостоятельно.

1.1.2. Первая программа

В Eclipse

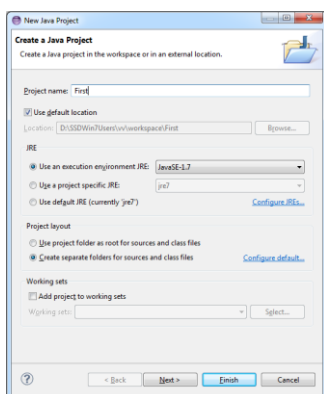
После первого запуска Eclipse (о построении проекта в IntelliJ IDEA написано ниже) мы увидим такое окно:



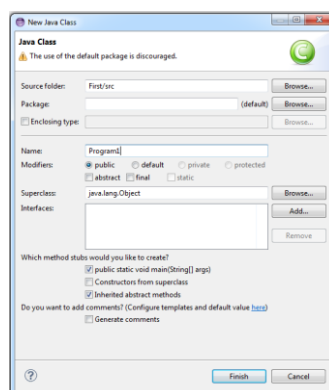
Workspace – это место, где располагаются все наши проекты. По умолчанию это папка workspace в домашней папке пользователя, но можно назначить любое другое место для хранения своих Java-проектов. **Запомните адрес в поле ввода** (у вас будет, конечно, не такой как на скриншоте). Он пригодится в дальнейшем, когда мы будем выбирать файл с программой для тестирования. Если вы уже не первый раз запускаете Eclipse, это окно может не появиться. Потом мы покажем, как можно узнать путь к workspace в уже работающей среде.

После запуска надо создать проект и класс:

File ⇒ New ⇒ Java Project



File ⇒ New ⇒ Class



При этом достаточно ввести только имена проекта и класса. Не используйте русских букв в названиях.

При создании класса лучше всего выбрать опцию `public static void main(String[] args)`, чтобы генератор создал главную функцию класса «за нас», а поле `package` оставить пустым (default), иначе программа не будет корректно работать в тестирующей системе.

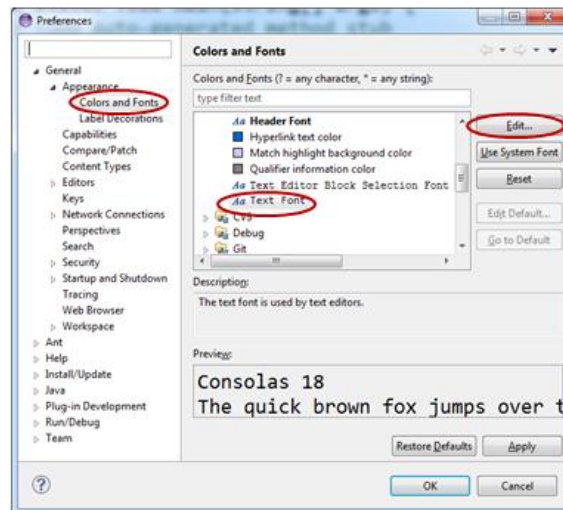
По умолчанию шрифт достаточно мелкий, возможно сразу стоит его подстроить «под себя»:

Window ⇒ Preferences

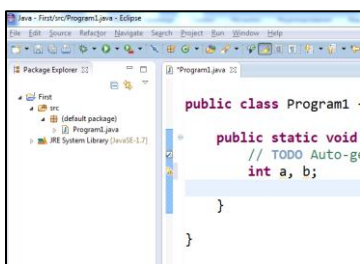
слева **General ⇒ Appearance ⇒ Colors and Fonts**

справа **Basic ⇒ Text Font**

кнопка **Edit...**



Приступаем к набору текста в основном окне:



Слева в окне – Навигатор. Открыт проект First в нем Java-файл Program1

Сразу приведем текст программы и прокомментируем важные части кода. Обучение программированию по примерам частый и хороший прием, но при этом надо стараться понимать все, что написано, каждую строчку.

Двумя слешами (//) начинаются комментарии – это текст, который компилятор игнорирует, а мы будем его использовать для объяснения важных моментов прямо в коде. Наверняка вы заметили, что генератор сам вставляет в создаваемый код комментарий

```
// TODO Auto-generated method stub
```

говорящий о том, что именно здесь, после него нужно писать программу. Его можно удалить.

Полный текст программы:

```
import java.io.PrintStream;

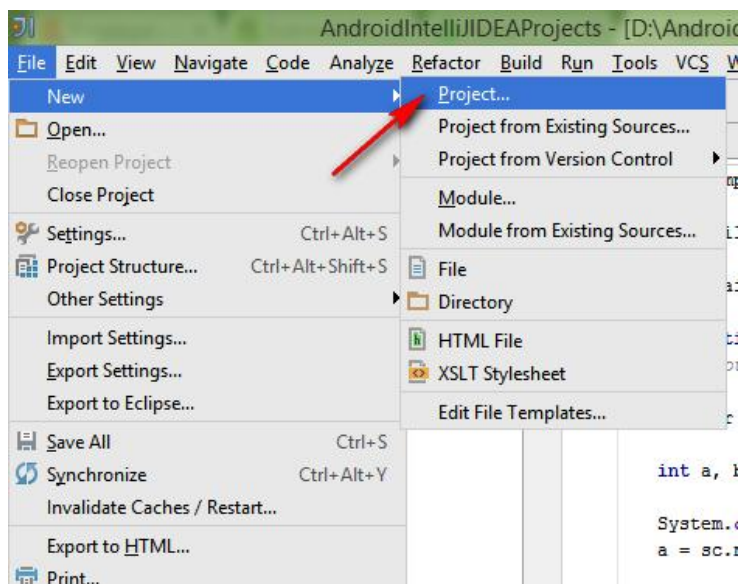
public class Program1 {
    public static PrintStream out = System.out;

    public static void main(String[] args) {
        out.println("Hello, world!");
    }
}
```

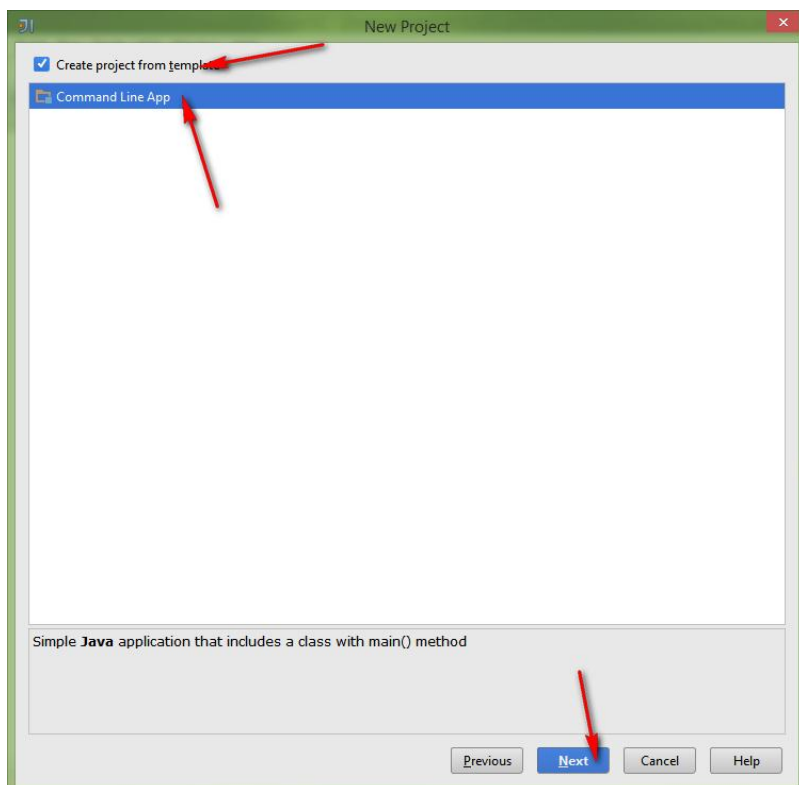
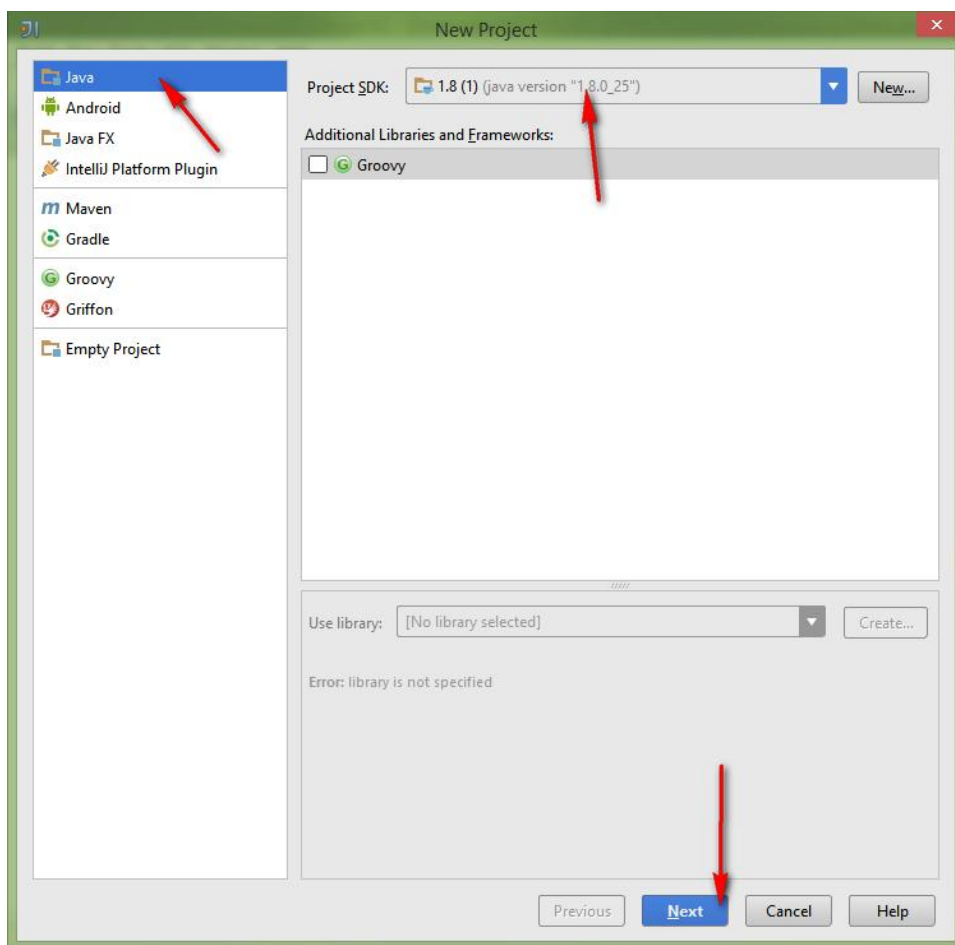
B IntelliJ IDEA

(Материал этого раздела подготовлен по статье преподавателя школы Samsung Антона Борисовича Сергиенко <http://blog.harrix.org/?p=3248>)

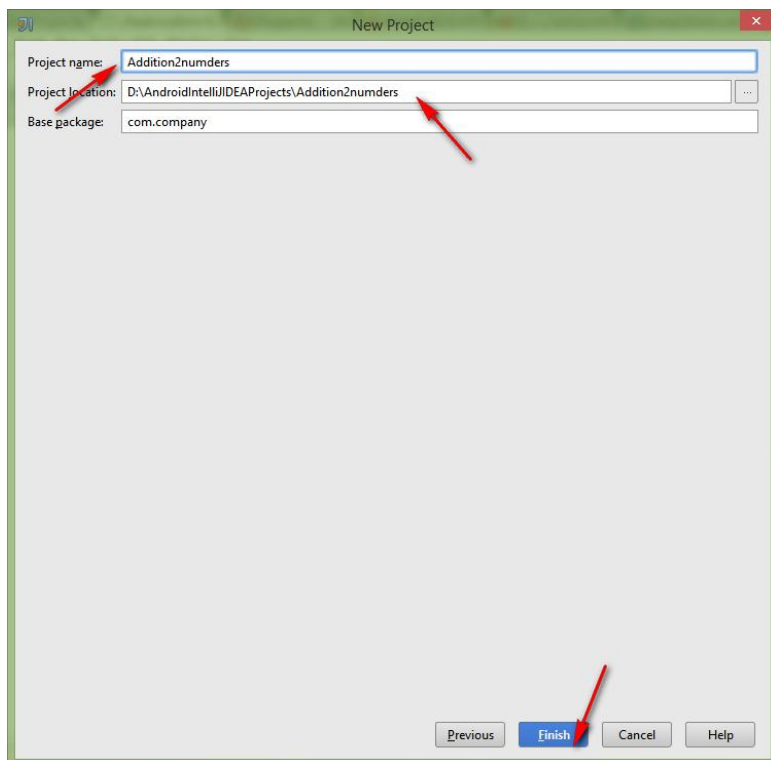
Открываем IntelliJ IDEA.



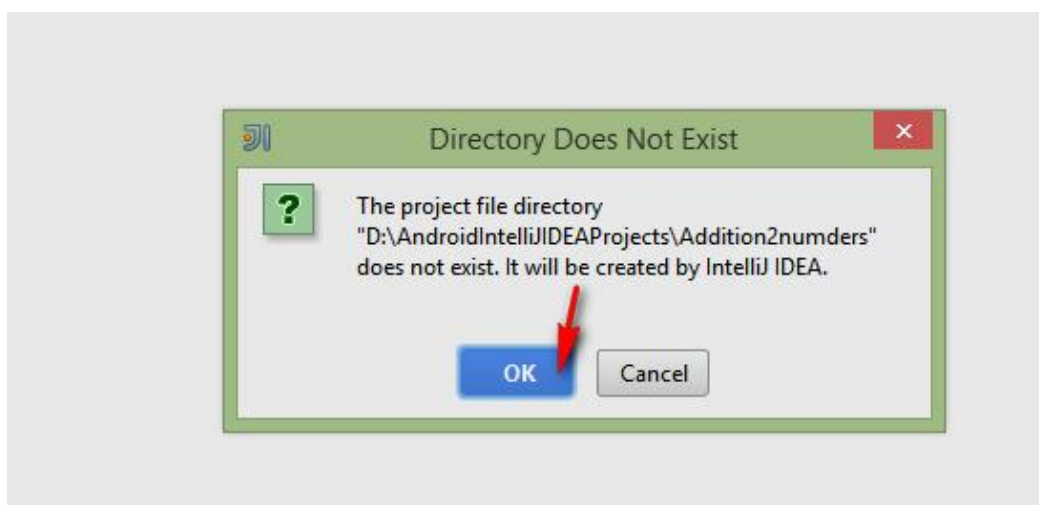
Далее



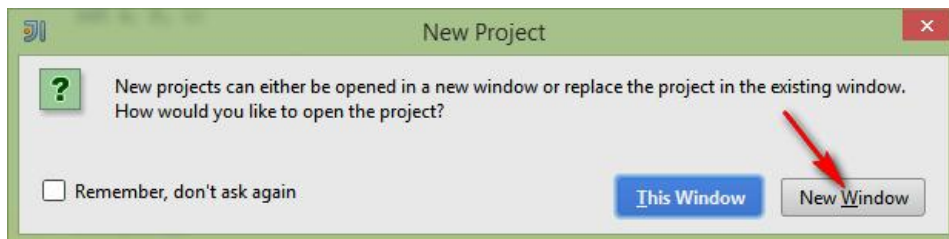
Нужно выбрать имя приложения (желательно без русских букв и пробелов) и папку размещения. Можно указать и свой домен, если таковой имеется, но для тестовых приложений это не важно.



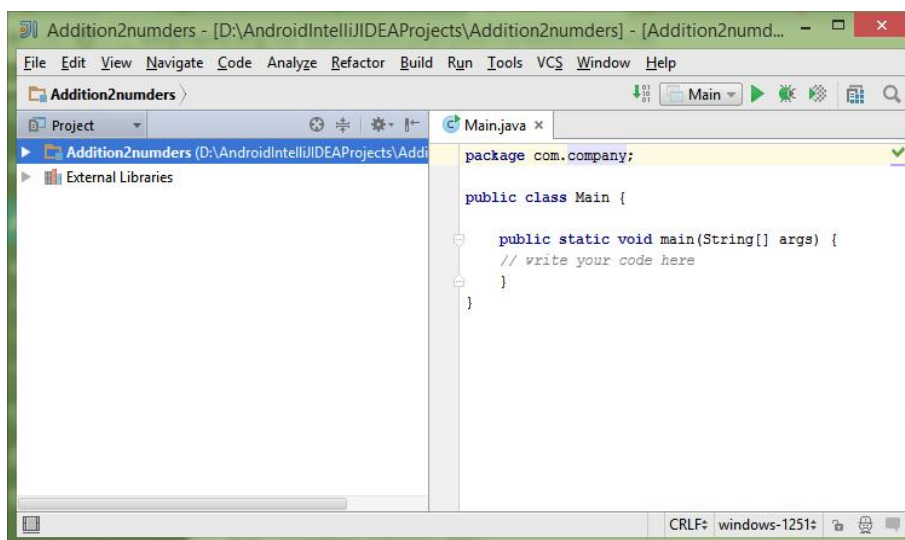
Если вы указали несуществующую папку, то он спросит у вас: создавать ее или нет.



Потом предложит в том же окне открыть, что и предыдущий проект или в новом. Лучше выбирать в новом.



И проект создан.



Программа «Hello, world» на Java замечательна тем, что на ее примере можно обсудить многие особенности программирования на этом языке.

Все слова, кроме, пожалуй, текста в кавычках, требуют осмысления.

Прежде всего, программа на Java это класс или несколько классов. Для простейшей программы достаточно одного класса, но он использует класс System, библиотечный, стандартный класс, который устанавливается вместе с компилятором Java.



Компилятор или интерпретатор?

Рассмотрим ответ на вопрос: как компьютер, который работает только с машинными командами, понимает текст, написанный программистом?



На выполнение Java-машине отдается именно класс. Для того чтобы Java-машина смогла выполнить класс, он должен быть объявлен с модификатором **public**. Имя класса при этом должно совпадать с именем файла с исходным текстом с учетом регистра, даже для *Windows*. При работе в *Eclipse* это происходит автоматически.

Класс может содержать в себе методы (функции, собственно, наборы команд) и поля (данные, в *Java* это примитивные типы и объекты – представители других классов). Говоря совсем просто, класс – это сущность, которая в себе что-то хранит и что-то умеет.

Разберем решение построчно.

Сначала создается инструмент вывода:

```
static PrintStream out = System.out;
```

Он объявлен в классе до функции **main**, что делает их видимыми для любой функции. Хотя у нас сейчас она одна – **main**, потом мы, возможно, захотим добавить другие функции в программу и они смогут пользоваться созданным инструментом.

Оператор **import** сообщает компилятору Java, где найти классы, на которые ссылается код. Любой сложный объект использует другие объекты для выполнения тех или иных функций, и оператор импорта позволяет сообщить о них компилятору Java.

Оператор импорта обычно выглядит так:

```
import package_name.subpackage_name.class_name;
```

За ключевым словом **import** следуют класс, который нужно импортировать, и точка с запятой. Имя класса должно быть полным, то есть включать свой пакет. Чтобы импортировать все классы из пакета, после имени пакета можно поместить *****.

Например, строку

```
import java.io.PrintStream;
```


можно заменить на

```
import java.io.*;
```

Тогда будут импортированы все классы из пакета **java.io**. В том числе **Scanner**, который нам понадобится на следующем занятии.

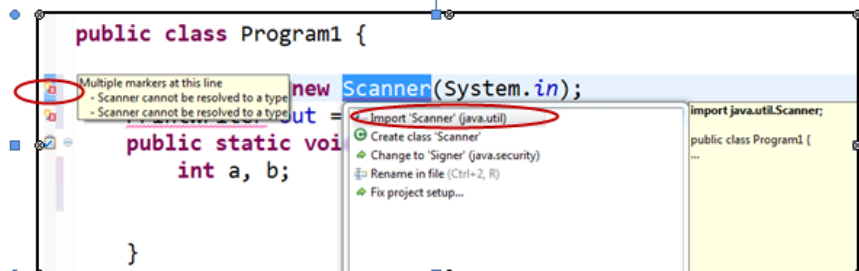
Директива **import** существенно отличается от **#include** в C++. Можно вообще не использовать **import**, а использовать полные имена классов (с именами пакетов). Например строка создания поля **out** типа **PrintStream** может выглядеть так:

```
static java.io.PrintStream out = System.out;
```



Если набирать программу, начиная с функции `main`, Eclipse сообщает подчеркиванием о том, что нужно добавить директиву `import` в программе, чтобы указать, в какой библиотеке лежат классы `Scanner` и `PrintStream`. Можно добавить эту директиву самостоятельно, но проще нажать на лампочку слева и выбрать предлагаемое средой решение щелчком мыши.

При этом нужно быть очень осторожным. Часто бывает, что классы с одинаковыми названиями лежат в разных пакетах, и нужно быть уверенными, что вы импортируете нужный класс.



Невозможно создать просто глобальную функцию, она должна принадлежать какому-то классу. Обычно для этого создается объект класса и из него вызывается функция. При старте Java-программы JVM вызывает функцию с именем `main` (имя зарезервировано, как и в C/C++) по имени класса без создания объекта. Для этого функция должна быть объявлена с модификатором `static`.

Функция ничего не возвращает, только выполняет печать текста. Тип возвращаемого значения в этом случае – `void`.

Функция `main` принимает массив объектов класса `String`. Квадратные скобки для объявления можно в Java ставить и до и после объявления типа. Строки, как и массивы в Java тоже являются объектами.

В строке вывода

```
out.println("Hello, world!");
```

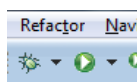
печать осуществляется посредством вызова функции `println` статического объекта `out` класса `PrintStream`.

Для вывода у `PrintStream` две функции: `print()` и `println()`. Вторая после вывода переводит курсор на новую строку. `PrintStream` представляет буферизованный поток.

Имена классов и интерфейсов (о них мы поговорим позднее) принято начинать с заглавной буквы, объектов и данных примитивных типов с малой.

Программа написана, можно ее запустить и проверить, как она работает.

Нажимаем белую стрелку в зеленом круге

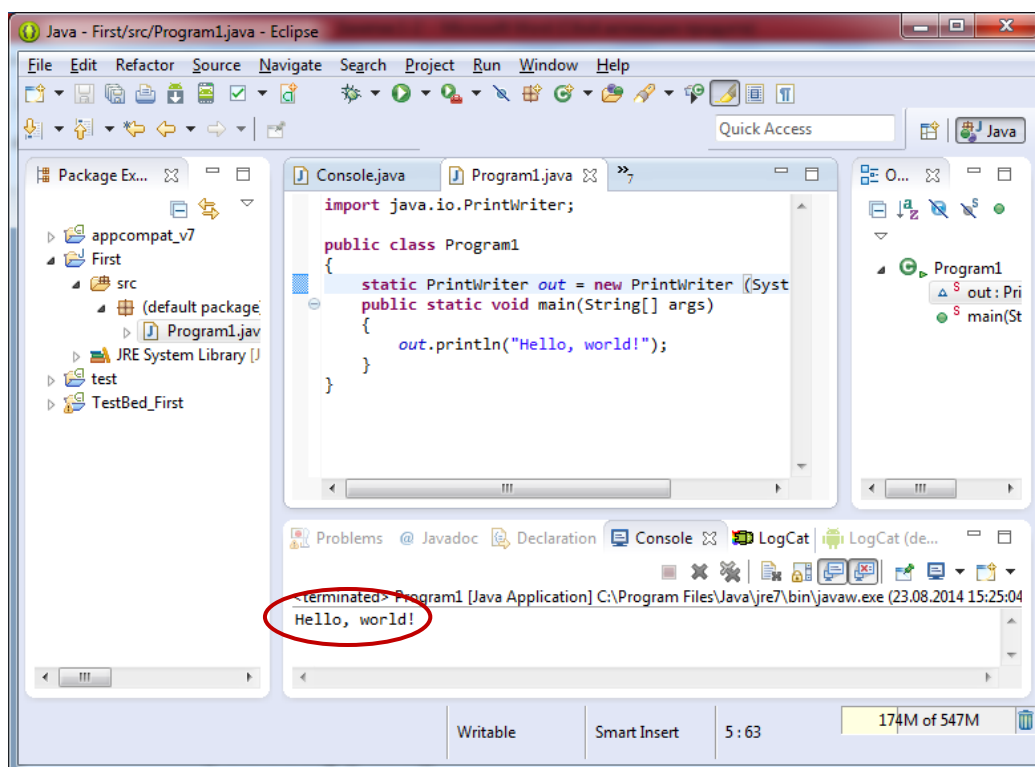


Обратим, кстати, внимание на кнопку-жучка. Она тоже очень важна – при помощи ее мы сможем исполнять программу по шагам (по командам) и с помощью этого находить и исправлять ошибки – это вызов **отладчика**.

Пора запускать программу!

Куда будет осуществляться вывод? «Обычный» System.out (объект потока-вывода) выводит информацию в консоль. На этом же занятии мы будем использовать другой объект и выведем информацию в окно мобильного устройства.

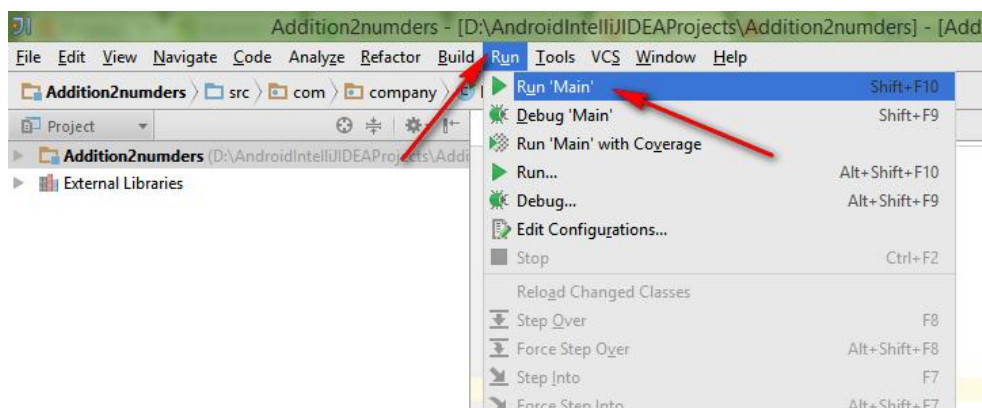
В Eclipse



Hello, world вывелось в нижнем правом углу экрана – на консоль Eclipse.

В IntelliJ IDEA

В IntelliJ IDEA запуск программы абсолютно аналогичен.





Важно понимать, как это происходит. На самом деле `PrintStream` использует переданный ему в конструкторе поток для вывода.

Сам поток может уметь выводить только один байт. `PrintStream`, выполняя команду `println`, разбирает переданную строку на символы и много раз командует своему потоку вывести каждый символ.



*Если внешний вид отличается от скриншота, и, возможно, некоторые важные элементы окна исчезли совсем, можно выбрать в меню **Window** ⇒ **Reset Perspective...** и при утвердительном ответе все восстановится.*

*Так же можно показывать-скрывать отдельные окна интерфейса через меню **Window** ⇒ **Show View***

1.1.3. Запуск на мобильном устройстве

Прежде всего, нужно подготовить мобильное устройство к загрузке программ.

Для этого

- на компьютере разработчика должны быть установлены драйвера для этого устройства;
- на самом устройстве должна быть включена отладка по USB (см. “Параметры разработчика” в Настройках¹);
- устройство должно быть подключено к компьютеру проводом USB.

Об эмуляторах мы поговорим чуть позже.



Очень Важное Замечание!

При работе с Android-проектами Eclipse заметно притормаживает. Нужно к этому очень терпеливо относиться. Например, сразу после загрузки Eclipse может показать много ошибок в проекте, но через полминуты все станет нормально.

Если при этом немедленно пытаться исправить ситуацию, щелкая мышкой по управляющим элементам среды, и фактически, запуская новые команды, скорее всего Eclipse зависнет совсем и его придется перезагружать.

Будьте терпеливы!

Если все это сделано... то все равно нельзя просто взять и запустить нашу программу на Android-устройстве. Для того, чтобы программа работала, нужно, чтобы она была написана специально под Android. Соответственно нужно создать не Java Project, а Android Application Project. И дальше написать программу, специально ориентированную под Android. Мы это обязательно сделаем в курсе, но сейчас воспользуемся нашей Samsung School-песочницей.

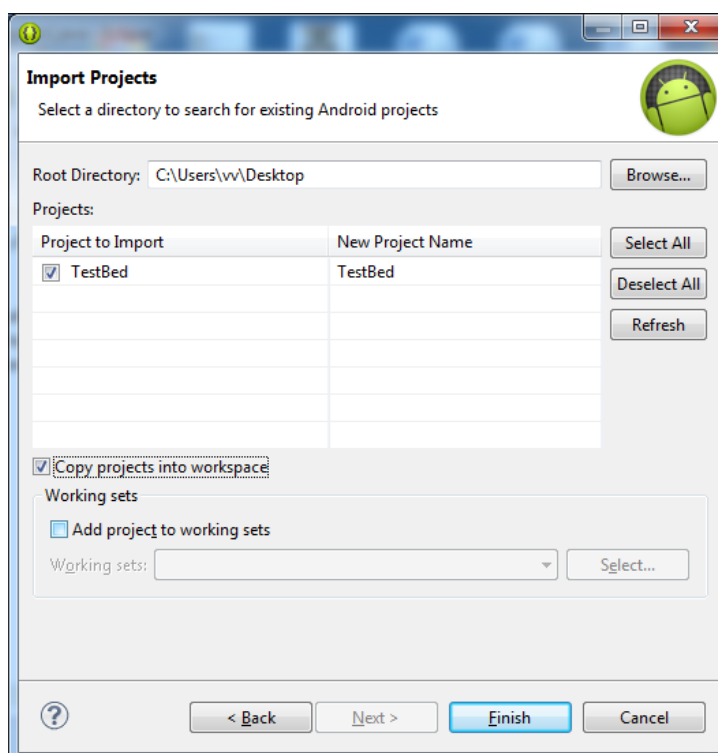
¹ На некоторых устройствах с ОС Android 4.2 и выше в настройках отсутствует этот пункт. Чтобы его открыть, найдите пункт «Об устройстве» (или «О телефоне») в нем нажмите на «Номер сборки» 7 раз подряд, после чего откроется меню “Параметры разработчика”.

Для упрощения работы на первых занятиях инженеры Samsung совместно с преподавателями школы написали Android-программу TestBed (в переводе «испытательный стенд»), в которую легко встроить наш код, ориентированный на работу с консолью и тогда ее можно будет запустить.

Ваша программа может запускаться в консольном режиме на многих операционных системах и в различных виртуальных машинах. TestBed это еще одна «виртуальная машина», в которой вы можете запустить свою программу.

Для дальнейшей работы скопируйте проект из указанной преподавателем папки на сервере или системы обучения и импортируйте его.

File ⇒ Import, в появившемся списке **Android ⇒ Existing Android Code into Workspace**

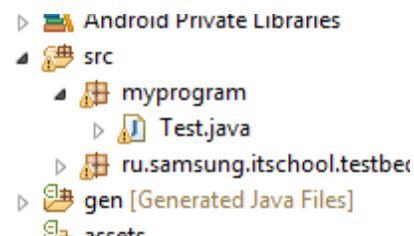


В появившемся окне желательно ставить флажок Add project to working sets, чтобы все проекты были в одном месте.

Мы рассмотрим внутреннюю структуру проекта TestBed позже.

Сейчас важно, что исходный код состоит из двух пакетов.

- Пакет myprogram с единственным файлом-классом MyProgram, именно его содержимое и будет выполнено на испытательном стенде.
- И пакет с длинным названием ru.samsung.itsc... - сам **TestBed**.





Важно, чтобы разные программисты имели возможность использовать одинаковые имена классов в своих проектах. Имя домена сайта разработчика уникально по определению, поэтому размещение файлов в пакете с таким именем гарантирует отсутствие конфликтов. При этом для удобства имена пишутся в обратном порядке. При этом естественным образом соблюдается иерархичность. Например, следующий проект IT школы Samsung при импорте автоматически расположится в папке ru/samsung/itschool, рядом с TestBed.

Для того, чтобы выполнить нашу программу на мобильном устройстве, нужно заменить содержимое класса MyProgram на содержимое нашего класса.

Полностью программа будет иметь такой вид:

```
package my_program;

import java.io.PrintStream;
import java.util.Scanner;

public class MyProgram
{
    public static Scanner in = new Scanner(System.in);
    public static PrintStream out = System.out;

    public static void main(String[] args) {
        out.println("Hello, world!");
    }
}
```

Мы видим, что программа практически ничем не отличается от первоначальной. В ней добавились только несколько строк. Имя пакета, которому принадлежит класс – наша программа и объявление входного потока, чтобы TestBed мог его переопределить. В нем статический объект *out* организует вывод символов на экран мобильного устройства. А для стандартного *PrintStream* это не важно – он «командует» «своему» потоку и тот выводит «как умеет».

Запуск программы происходит стандартным образом, но программа стартует уже на устройстве!

И последнее, может быть не самое интересное, но важное. Даже здесь, в проекте *TestBed* можно запустить программу обычным образом, как в начале занятия - в обычной консоли.

Программа на *Java* состоит из нескольких классов. Если мы запускаем *TestBed* на Android - запускается внутренняя программа, которая в процессе своей работы выполняет функцию *main* из класса *MyProgram*. Но можно запустить и непосредственно класс *MyProgram*. Для этого нужно выбрать конфигурацию запуска (по имени конфигурации или **Run As Java Application.**) В выданном вам проекте уже может быть настроены оба способа запуска.

Задание 1.1.1

Установите пакет разработчика IT Samsung School на домашний компьютер, напишите и запустите первую программу:

- Выполните импорт проекта TestBed и попробуйте запустить программу на мобильном устройстве.
- Технические неудачи подробно зафиксируйте (перепишите дословно появившиеся сообщения об ошибках или сохраните скриншоты). Это позволит преподавателю быстрее помочь разобраться с возникшей проблемой на следующем занятии.

Благодарности

Компания Samsung Electronics выражает благодарность за участие в подготовке данного материала преподавателям ИТ ШКОЛЫ SAMSUNG Ильину Владимиру Владимировичу и Сергиенко Антону Борисовичу.

.