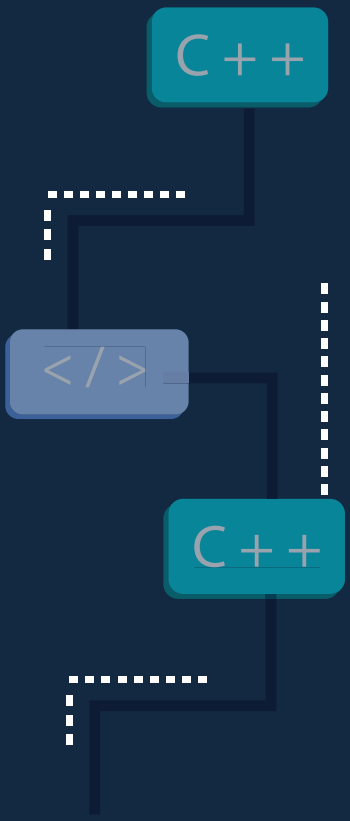
top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ



Язык программирования JAVA

Урок №4

Циклы

Содержание

1. Циклы

Для реализации циклического повторяющегося алгоритма в Java есть операторы **while**, **do while** и **for**.

while – оператор цикла «пока не».

Синтаксис:

```
while (выражение) операция;
```

Выражение в скобках (далее условие выполнения цикла) должно иметь тип boolean. В случае если условие выполнения цикла принимает значение истина (true), будет выполнена «операция» и управление передастся обратно в проверку выражения. Если значение ложно, выполнение тела цикла прекратится.

Пример:

```
int i = 0;
while (i < 5) System.out.println(i++);
```

Результат: 0, 1, 2, 3, 4, 5

Телом цикла называется операция или блок кода, ограниченный фигурными скобкам и находящийся сразу после оператора while.

Пример:

```
int i = 0;
while (i < 5) { // начало тела цикла
 i++;
 System.out.println(i);
} // конец тела цикла
```

Результат: 1, 2, 3, 4, 5

Примечание. *Если условие выполнения цикла изначально ложно, тело цикла ни разу не будет выполнено.*

Пример:

```
int i = 0;
while (i > 5) {
 System.out.println(i++);
}
System.out.println(i);
```

Результат: 0

Если члены условия цикла не меняются внутри тела цикла, а условие выполнения цикла истинно, такой цикл будет выполняться бесконечно.

Пример бесконечного цикла:

```
while (true) {
 System.out.println("infinity");
}
```

do-while – цикл с постусловием

Синтаксис:

```
do операция;
while(выражение);

do {
 операция1;
 операция2;
}
while(выражение);
```

Отличие **do while** от цикла **while** состоит в том, что тело цикла **do while** гарантированно выполнится один раз, до проверки условия выполнения цикла. Повтор-

ное выполнение тела цикла будет зависеть от значения выражения, если оно истинно, то программа передаст управление в начало тела цикла.

Пример:

```
int i = 0;
do
{
 i++;
 System.out.println(i);
} while (i > 10);
```

Результат: 1

for – выполнить цикл «n-раз»

Синтаксис:

```
for (<инициализация>; <условие>;
      <счетчик>) операция;
```

Блок «инициализация» может содержать объявление и инициализацию переменных, которые будут доступны в теле цикла. Заполнение блока не обязательно.

Блок «условие» должен содержать выражение результатом которого должно быть значение типа boolean. Заполнение блока не обязательно.

Блок «счетчик» может содержать любые операции. Заполнение блока не обязательно.

Пример все блоки заполнены:

```
for (int i = 0; i < 10; i++)
{
 System.out.println(i);
}
```

Пример с отсутствием объявление переменной счетчика (необходим, если нужен доступ к счетчику за пределом тела цикла):

```
int i;
for (i = 0; i < 10; i++)
{
 System.out.println(i);
}
i+=2;
System.out.println(i);
```

Пример с отсутствием блока инициализации:

```
int i = 0;
for (; i < 10; i++)
{
 System.out.println(i);
}
```

Пример с использованием нескольких счетчиков:

```
for (int i = 1, j = 3; i < 5 & j > 0; i++, --j)
{
 System.out.println("i = " + i + " j = " + j);
}
```

Результат: i = 1 j = 3, i = 2 j = 2, i = 3 j = 1

for-each

В Java нет специального колючего слова для перебора элементов массива или коллекций, как в других языках. Для реализации цикла for-each используется специальный синтаксис оператора for.

Синтаксис:

```
for (тип элемент : коллекция) {
        операция;
}
```

Пример для массива:

```
int [] numbers = new int[] { 3, 2, 1 };
for (int number : numbers)
{
 System.out.print(number + " ");
}
```

Пример для коллекции:

```
List<Integer> lists = new ArrayList<Integer>();
lists.add(1);
lists.add(3);
for (Integer value: lists)
{
 System.out.println(value); }
```

Операторы break и continue

Оператор **break** прерывает выполнение цикла и передает управление в конец тела цикла.

Пример:

```
int i = 0;
while (i < 10)
{
 System.out.println(i);
 if (i >= 2) break;
 i++;
}
```

Результат: 0, 1

Оператор **break** может быть использован совместно с меткой, например для одновременного прерывания нескольких вложенных циклов. В этом случае управление будет передано в конец блока помеченного меткой указанной после **break**. Метка – это именует блок кода.

Пример:

```
two: for (int n = 0; n < 10; n++) {
   for (int j = 10; j > 0; j--) {
   System.out.print(j - n + " ");
   if (j + n == 5 && n > 0)
   break two;
   }
  }
```

Оператор **continue** прерывает выполнения тела цикла и передает управление в начало цикла. Происходит пропуск текущей итерации цикла.

Пример:

```
int i = 0;
while (i < 10)
{
 i++;
 if (i % 2 == 0)
 continue;
 System.out.println(i);
}
```

Результат: 1, 3, 5, 7, 9

Оператор **continue** так же может применяться с меткой.

Пример:

```
one: for (i = 0; i < 10; i++) {
  for (int j = 0; j < 10; j++) {
   if (i == j) continue one;
    System.out.printf("i=%d j=%d \n", i, j);
  }
  }
```

Оператор return

Оператор **return** прекращает выполнение метода, возвращая управление в место вызова метода.

2. Основные положения Java Code Convention

Code Conventions for the Java Programming Language – это соглашение по оформлению кода. Соблюдение положений конвенции делает Ваш код более читаемым и понятным для других программистов. Эти положения являются рекомендациями, которые составлены на основе личного опыта при работе с кодом программистов.

Имена классов

Фалы с исходным кодом должны иметь расширение java.

Файлы с байт-кодом должны иметь расширение class.

Организация файла исходного кода

Размер файла не должен превышать 2000 строк. Файлы с большим количеством строк затрудняют поиск нужного блока кода, увеличивают время компиляции и время загрузки класса в Java машину.

Файл должен разделяться на секции, которые отделяются друг от друга пустой строкой.

В одном файле может находится только один класс с модификатором public (основной класс).

Не желательно размещение в одном файле других классов (не public).

Порядок следования секций в классе:

- javadoc;
- заголовок класса;
- статические поля;
- нестатические поля;

- конструкторы;
- методы.

Отступы

Единица отступа – 4 пробела (может использоваться табуляция).

Длина строки – не более 80 символов.

Именование классов и интерфейсов

Классы должны именоваться с большой буквы, английскими символами без пробелов. Имя класса должно соответствовать одному объекту этого класса.

Пример:

```
class Cats // плохо
class Cat // хорошо
```

Если в имени класса содержится более одного слова, каждое слово выделяется большой буквой (Camel-нотация). Имя класса должно соответствовать.

Пример:

```
class TileIterator
```

Интерфейсы должны именоваться по правилам классов. Не желательно использование дополнительных указаний в имени интерфейса на принадлежность к интерфейсу.

```
interface IClonable // плохо
interface Clonable // хорошо
```

Если в имени класса содержится аббревиатура содержащая больше двух букв, все буквы аббревиатуры кроме первой должны быть строчными.

Пример:

```
class FPSRenderer // плохо
class FpsRenderer // хорошо
```

Именование переменных

Объявление (декларация) нескольких переменных должно быть в разных строках.

```
int a, b; // плохо

int a; // хорошо
int b; // хорошо
```

Если возможно инициализировать переменные в месте объявления.

```
int a;
...
a = 255; // плохо

int a = 255; // хорошо
```

Необходимо объявлять переменные как можно ближе к месту использования.

```
int a = 255; // плохо
System.out.println("file")
a + = 127;
```

```
System.out.println("file")
int a = 255; // хорошо
int c = a + 127;
```

Массивы

При объявлении массива необходимо указывать скобки после указания типа переменной, а не после имени.

```
int a [] = new int[3]; // плохо
int[] a = new int[3]; // хорошо

int [] a [] = new int[3]; // плохо
int[][] a = new int[3][3]; // хорошо
```

Константы

Константы принято именовать большими буквами, а в качестве разделителя использовать символ подчеркивания.

Примеры:

```
final int MAX_STEP = 3;// хорошо
final String DELIMITER = «;»;// хорошо
```

Методы

Методы должны именоваться латинскими буквами, начиная с маленькой буквы. Первой слово в названии метода должно быть глаголом или наречием. Если в имени метода содержится более одного слова, каждое слово кроме первого выделяется большой буквой (Camel нотация).

Примеры:

```
void startprocess() {} // плохо
void Start() {} // плохо
void start_process() {} // плохо
void sProcess() {} // плохо
void sp() {} // плохо
void start() {} // хорошо
void startProcess() {} // хорошо
```

3. Работа с интегрированным отладчиком в Eclipse

Eclipse (*от англ. затмение*) – свободная интегрированная среда разработки модульных кросс-платформенных приложений. Развивается и поддерживается Eclipse Foundation.

Альтернативные среды разработки

Таблица 1. Рейтинг Java-IDE, используемых в настоящее время

N	Название	% голосов	Оценка
1	Eclipse	19.77	4.6
2	IntelliJ IDEA	19.06	4.7
3	NetBean	7.11	4.1
4	JBuilder	5.68	4.2
5	JDeveloper	2.13	4.0
6	JCreator	1.70	3.9

Таблица 2. Рейтинг ранее использовавшихся Java-IDE

N	Название	% голосов	Оценка
1	Eclipse	21.47	3.0
2	IntelliJ IDEA	16.64	3.3
3	NetBean	14.22	2.9
4	JBuilder	11.66	3.5
5	JDeveloper	7.11	2.8
6	Visual J++	5.26	1.8
7	JCreator	4.26	2.3
8	VisualAge for Java	3.69	2.8
9	JCreator	3.41	2.0

Установка Eclipse

Со страницы http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/indigor скачиваем Eclipse Indigo, с учетом типа и разрядности операционной системы.

Скачиваем архив. Распаковываем в удобное место: **Eclipse** не устанавливается через инсталлятор, он *portable*, в отличие от того же **Netbeans**.

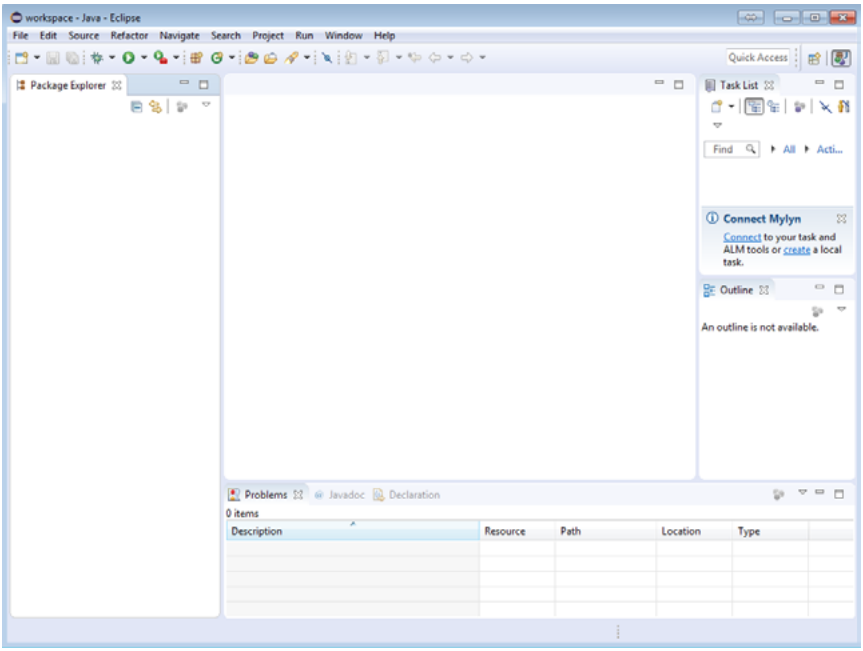
Создание проекта в Eclipse

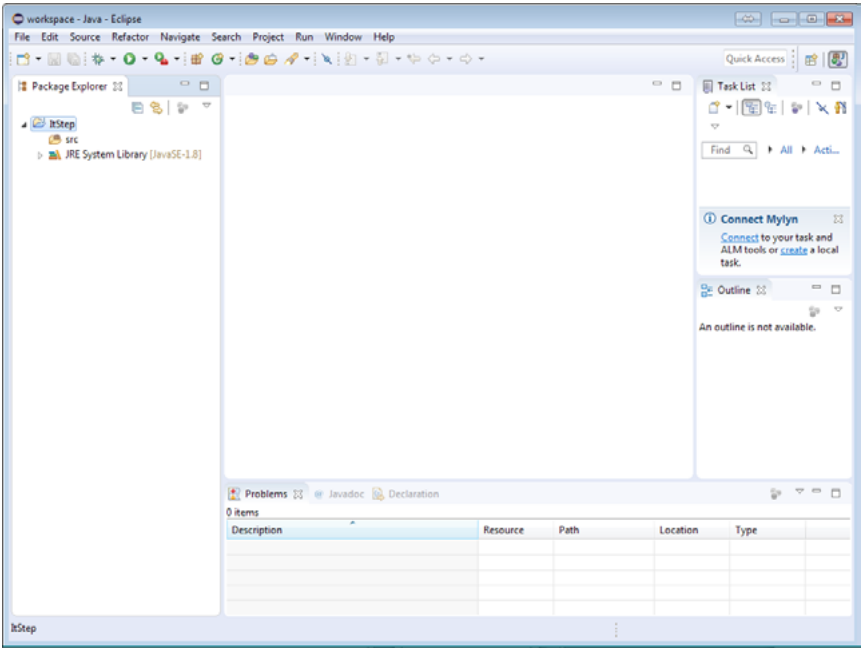
Запустим Eclipse SDK и убедимся, что открыта проекция Java.

В открывшемся диалоговом окне введем имя проекта. Располагаться проект будет в директории, установленной как "Workspace" при настройке Eclipse.

В следующем диалоге перейдем на вкладку "Libraries". Здесь пока ничего менять не будем, но запомним, что на этой вкладке можно добавить к проекту дополнительные Java библиотеки, а на данный момент к проекту подключен стандартный API, поставляемый с JRE.

Нажимаем "Finish". Создан пустой проект, для продолжения работы нужно добавить пакеты и классы.


workspace - Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
Task List
Find All Acti...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems Javadoc Declaration
0 items
Description Resource Path Location Type


workspace - Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
ItStep
src
JRE System Library [JavaSE-1.8]
Task List
Find All Acti...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems Javadoc Declaration
0 items
Description Resource Path Location Type
ItStep

Создадим пакет через контекстное меню. Кликнем правой кнопкой мыши на папке "src" и выберем **"New"** -> **"Package"**. То же можно проделать, нажав кнопку **"New Java Package"** на панели инструментов.

В появившемся диалоге введем имя пакета, оно должно быть уникальным во избежание коллизий имен. Как правило, разработчики инвертируют имя своего домена, вы тоже можете так поступить.

Создадим класс через контекстное меню. Кликнем правой кнопкой мыши на пакете и выберем **"New"** -> **"Class"**. То же можно проделать, нажав кнопку **"New Java Class"** на панели инструментов.

В диалоговом окне создания класса введем его имя, оно может быть любым, согласно правилам именования классов в Java. Отметим опцию "public static void main(String[] args)", тем самым укажем IDE создать для нас одноимённую функцию. Нажимаем "Finish".

Теперь справа мы видим структуру проекта и наш файл класса с расширением *JAVA*. По центру – исходный код класса, а справа – браузер классов, показывающий структуру пакетов и классов в виде дерева.

Отредактируем исходный код, введем инструкцию для вывода строки на консоль – System.out.println("Ваша строка"); . Не забываем пользоваться подсказками и автодополнением – начинаем вводить код и жмём **Ctrl + Пробел**. Не забудьте завершить строку символом ";".

Сохраняем изменения нажатием клавиш **Ctrl + S**.

Запуск Java проекта в Eclipse

Чтобы проверить работоспособность нашей программы, нажмем кнопку **"Run"** на панели инструментов или через главное меню. При первом запуске нужно выбрать запускать программу как обычное приложение или как апплет.

Выберите пункт "Java Application".

Ваше первое консольное Java приложение будет скомпилировано и выполнено. В открывшемся представлении "Console" в нижней панели главного окна IDE мы увидим вывод программы, а именно – нашу строку. Скомпилированные файлы классов с расширением *CLASS*, можно найти в папке с проектом -> "bin".