Java
(основы)

Sun's Description....
Java is a :

- Simple
- Object Oriented
- Distributed
- Interpreted
- Robust
- Secure
- Architecture Neutral
- Portable
- High Performance
- Multithreaded and Dynamic

Простой

- Has a small set of Language Constructs.
- Borrows the C and C++ syntax.
- Is free from pointers.
- Uses Garbage Collection.
- Does not use header files and preprocessors.

Объектный

- Not hybrid like C++.
- Supports the basic notions of OO :
  - Abstraction,
  - Modularity,
  - Encapsulation,
  - Hierarchy,
  - Typing,
  - Concurrency,
  - Persistance.
- Almost Everything is an Object.

Распределённый

- Works on a variety of platforms.
- Provides support for :
  - Networking,
  - Internet,
  - Remote Objects.

Интерпретируемый

- The Java Compiler generates bytecode for a JVM.
- A Java Interpreter is needed to execute the bytecode.

Надёжный

- Exception and Error handling.
- Multi-Tasking.
- Memory protection and management.
- Allows Modular development.
- Extensive compile-time checking.

Безопасный

- The features of bytecode and its interpretation, prevent unintentional or intentional sabotage of compiled programs.
- Security has been considered in many levels.

Нейтральный и переносимый

- Bytecode can run on any JVM on any platform.
- “Write Once run Anywhere”.
- JDK implementation on many platforms.

Производительный

- Multithreading allows more than one task in a program.
- With JIT compilers the interpreted code compiles at run time and gives almost native code speed.

Динамичный

- Java has been built to support the development of dynamically extendable systems.
- Objects can live on the internet.
- Java provides dynamic linking of the binary code at runtime.

Типы JAVA программ

- Аплеты и сервлеты
- GUI приложения
- Java Beans
- EJB

Что надо для программирования?

- JVM
- JRE
- JDK
- IDE
- Практика….

Основы синтаксиса

- Case sensitive.
- Each statement finishes with ';'.
- To begin and end a block you use '{' and '}'.
- Space, tab, and enter characters can be used to make the code more readable.

Комментарии

1. Comment on rest of the line :

```
// rest of this line is a comment
```

2. Multiple line comments :

```
/* This is a multiple line comment
   like in C or C++
*/
```

3. JavaDoc Comments :

```
/**
 * This comment will be included
 * in documentation
 * generated with 'javadoc'.
 */
```

Идентификаторы

Valid: 

```
stack,
Stack,
STACK_SIZE
wav2snd,
_snd,
$snd
```

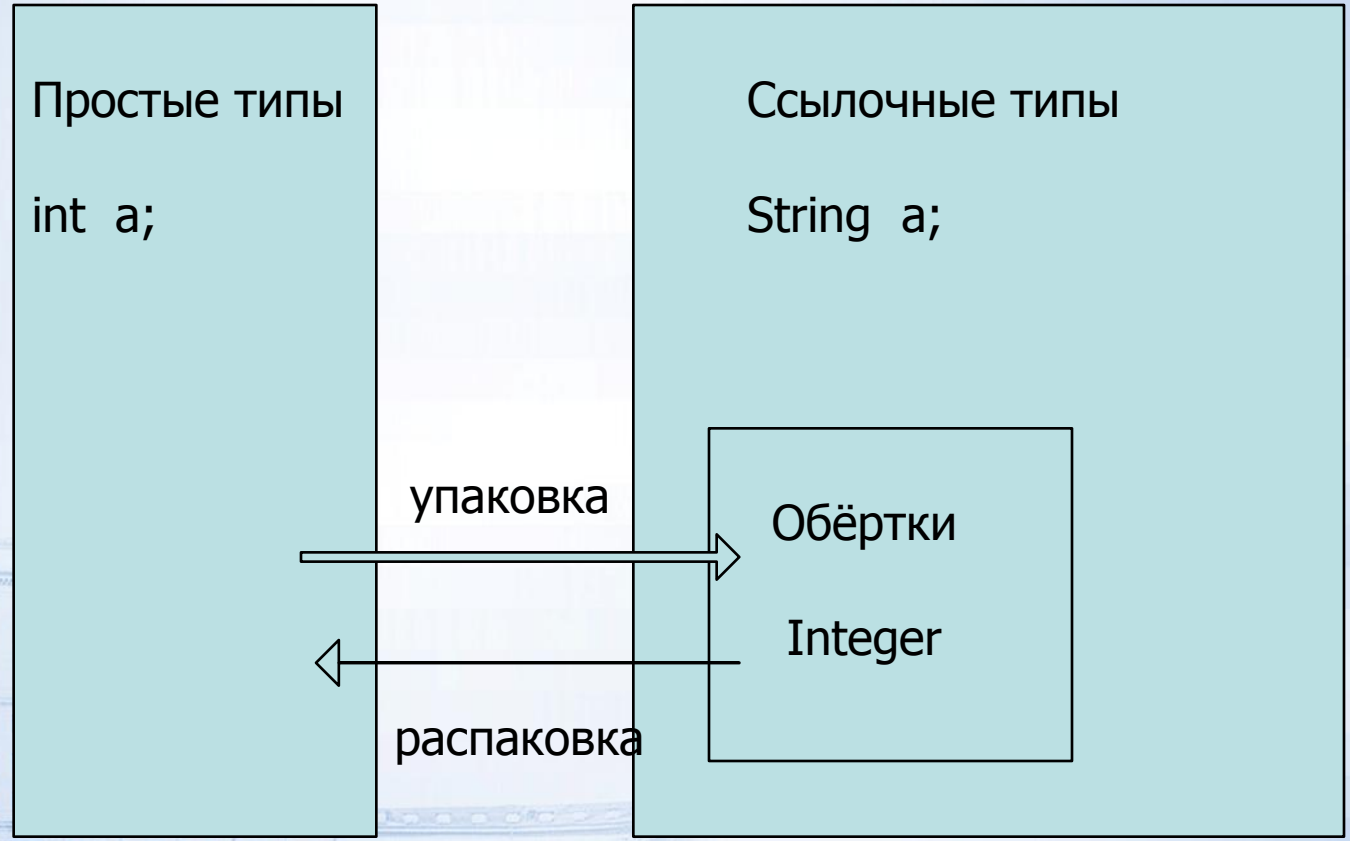
Invalid: 

```
3d,
5$,
#snd,
snd:wav
```

Ключевые слова

abstract	double	int	super
boolean	else	interface	switch
break	extend	long	synchronized
byte	false	native	this
byvalue	final	new	threadsafe
case	finally	null	throw
catch	float	package	transient
char	for	private	true
class	goto	protected	try
const	if	public	void
continue	implements	return	while
default	import	short	
do	instanceof	static	

Система типов



Type	Data Storage	Example
byte	signed 8-bit	$-2^{7} \ldots 2^{7}-1$
short	signed 16-bit	$-2^{15} \ldots 2^{15}-1$
int	signed 32-bit	$-2^{31} \ldots 2^{31}-1$
long	signed 64-bit	$-2^{63} \ldots 2^{63}-1$
boolean	1 bit	true or false
float	32-bit (IEEE-754)	3.4e+38
double	64-bit (IEEE-754)	1.7e+308
char	16-bit (unicode)	'\n'

Объявление переменных

```
class VarDecTest  {
        int   a, b = 0;
        long  millisec = 322245;
        char  cr = '\r';
        boolean probe = true;

         public static void main(String argv[]) {
           int sum = 10;
           long square = sum * sum;
          }
} //end class
```

Константы

- Numeric
  - long color = 0x12345 ; // hexadecimal
  - int register = 03744 ; // octal
- Character
  - char c = 'q';
- Boolean
  - true, false
- Object
  - null

Non Printing Characters

'\n'	newline
'\r'	return
'\t'	tab
'\b'	backspace
'\''	single quote

Преобразование типов

Type Casting = To change a basic type into another

```
int my_int = 70;
char c = (char) my_int;
```

* Casts that results in no loss of information :

From Type	To Type
byte	short, char, int, long, float, double
short	int, long, float, double
char	int, long, float, double
int	long, float, double
long	float, double
float	double

Operators	Precedence
postfix	*expr*++ *expr*–
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Методы

- Similar syntax to c++
  - <modifiers> <return_type> <name> (<args>) <block>
    - modifiers : public, private, protected, static, final
    - return_type : any type and void for nothing
    - name : Identifier - Method's name
    - args : Argument list

Механизм передачи аргументов

- Простые типы

  - by Value :
    - the argument may not be changed by the method called
      - The 8 datatypes are passed this way

- Ссылочные типы

- Перегрузка функций
  - void solve(int a)
  - void solve(ina, int b)
- Функции с переменным числом аргументов
  - void vsolve1(Object … arg)
  - void vsolve2(int [] … arg)

Операторы Java

- Выражение

  a+b/5

  count = count +1

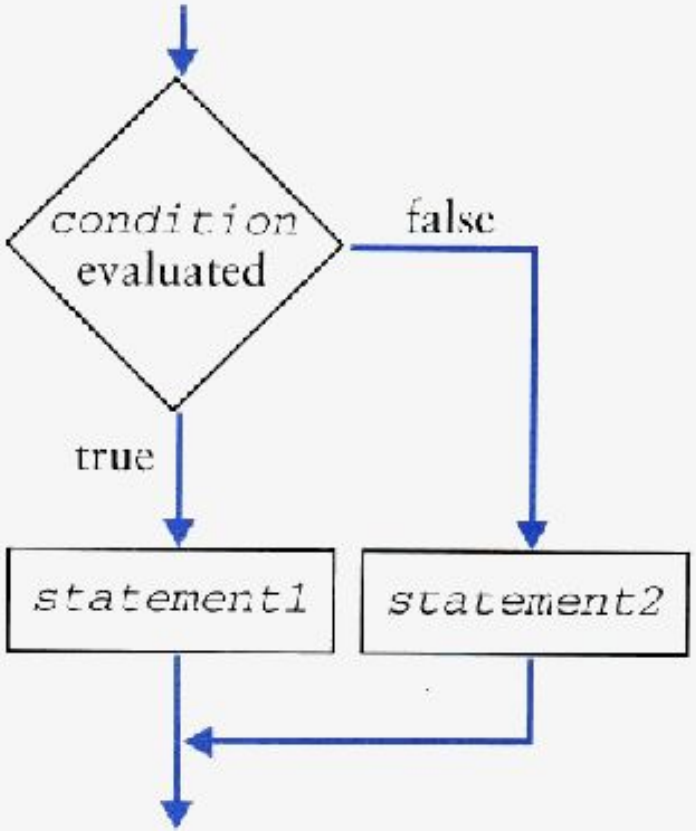
- Пустой оператор

  ;

- Блок

  { }

Branching - if - else



Syntax:

```
if ( boolean-expression )
  statement1;
[ else
  statement2; ]
```

Example:

```
boolean full(int fuel) {
  if (fuel < 80)
    return true;
  else
    return false;
}
```

Branching - switch-case-default

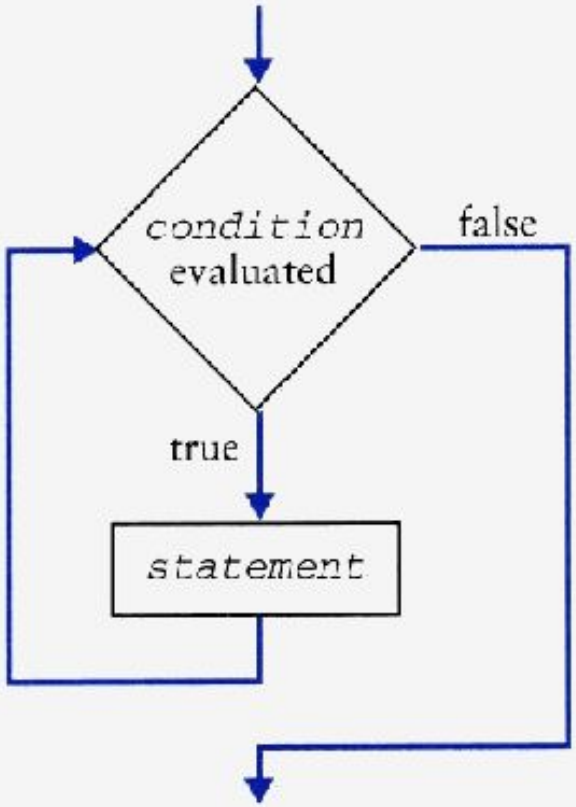
Syntax:

```
switch ( int-value ) {
        case int-value1:
          break;
        case int-value2:
          break;
        default:
      }
```

Example:

```
String numToString(int num) {
   switch (num) {
           case 1:  return "one";
           case 2:  return "two";
           ……..
           case 50: return “fifty”;
           default: return "many";
    }
}
```

Looping Constructs, while



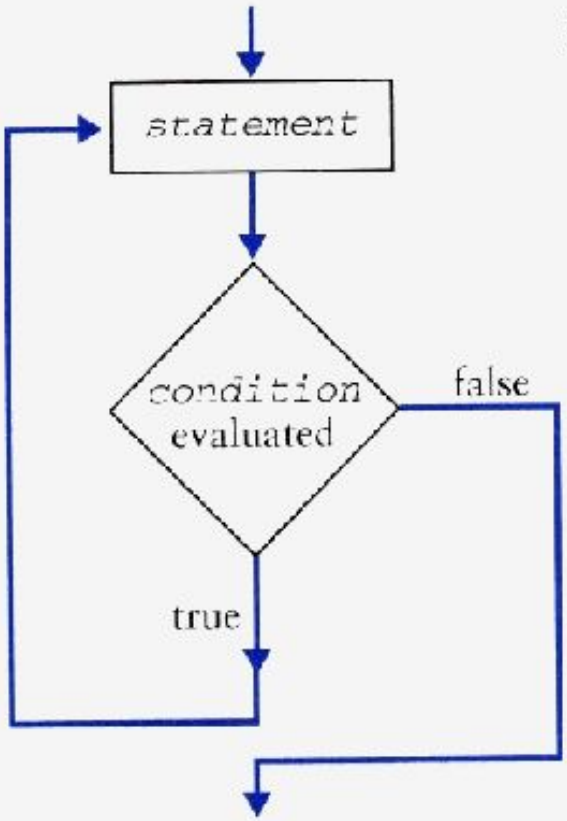
Syntax:

```
[initialization]
while ( boolean-expression ) {
  [statements]
  [iteration]
}
```

Example:

```
public static void main(String[] argv)
{
    int n = 0;
    while(n < argv.length) {
        System.out.println(argv[n]);
        n++;
    }
}
```

Looping Constructs, do-while



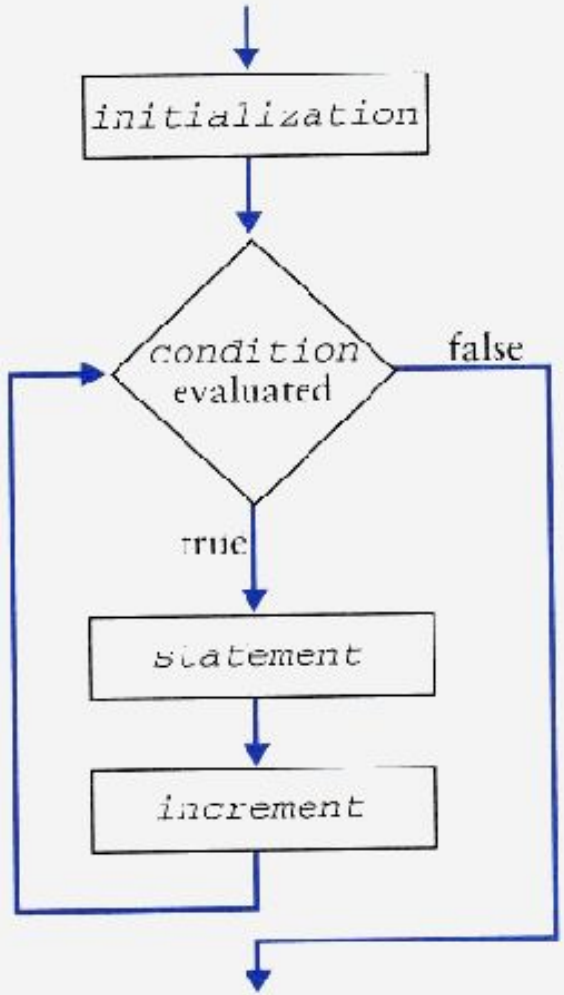
Syntax:

```
[initialization]
do {
  [statements]
  [iteration]
} while ( boolean-expression )
```

Example:

```
public static void main(String[] argv) {
  int n = 0;
  do {
    System.out.println(argv[n]);
    n++;
  } while (n < argv.length)
}
```

Looping Constructs, for



Syntax:

```
for ([initialization];
     [ boolean-expression ];
     [iteration]) {
  [statements]
}
```

Example:

```
public static void main(String[] argv) {
  for(int i = 0; i < argv.length; i++) {
    System.out.println(argv[i]);
  }
}
```

Цикл для коллекций

```
for( String s : argv)
    System.out.println(s);
```

Labeled blocks and branching

There is no 'goto' statements in Java, instead we have labeling.

```
a: {
b:   {
       if (t)
         break a;              // labeled break
       }
     .....
     }
```

Arrays in Java

- Arrays in Java are considered to be objects.
- Arrays can contain any simple or complex type.
- C++ like declaration : type t[];
- Java declaration : type[] t;
- Multidimensional Arrays : type[][]..[] t;
- Subscripts begin at zero.
- Out of bounds access *throws* an Exception.

- Объявление
  - int d[];
  - int d2[][];
- Создание
  - d = new int[10];
- Инициализация
  - int d[]={1,3,6};

Using Arrays

```
public class InitArrays {

   public static void main (String[] argv) {
     int [] numArray;          // declaration
     numArray = new int[10]; // creation
     for (int x=0; x<numArray.length; x++) {
        numArray[x] = x * 2;
     }
     for (int x=0; x<numArray.length; x++) {
       System.out.println("position " + x +
                          " contains: " + numArray[x] );
     }
   }
}
```

The String Object

- String, is a container type for 16-bit Unicode chars.
- Declaration : String str;
- Creation : str = new String("hi");
- Usage : String's methods, operator overloading, String-number conversions.

String's Methods

- Many, the most widely used :
  - `length()`
  - `equals(String)`
  - `startsWith(String)`
  - `toUpperCase()`
  - `toLowerCase()`
  - `indexOf(String)`
  - `subString(int begin)`
  - `subString(int begin, int end)`
  - `charAt(int index)`

The String + Operator

- The Operator + is said to be overloaded with respect to Strings.
- It can be combined with basic numeric types which then automatically convert to Strings.

```
String h = new String("Hello");
String t = new String(" there");
String ht = h + t ; // ht = "Hello there"
String ht2 = ht + 2 ; // ht2 = "Hello there2"
```

Converting Strings

- Any simple numeric to String
  - a. using String's `valueOf`(numeric)
  - b. using Number Classes and `toString()`
- String to numeric
  - ~~a. using String's `intValue()` or `floatValue()`~~
  - b. using Number Classes and `parse`(String)

- String - константные строки
- StringBuffer - thread-safe
- StringBuilder - изменяемая строка
- StringTokenizer – разбиение строки

```
StringTokenizer st = new StringTokenizer
                        (" this  is a  \n  test  ");
while (st.hasMoreTokens())
                System.out.println(st.nextToken());
```

- Регулярные выражения

```
String[] result = "this is a test".split("\\s");
```

- http://ru.wikipedia.org/wiki/Java
- http://darkraha.com/rus/java/
- http://www.linkex.ru/java/
- http://www.intuit.ru/department/pl/javapl/