```
/* пример # 7 : сравнение строк и объектов : ComparingStrings.java */
package chapt01;

public class ComparingStrings {
     public static void main(String[] args) {
          String s1, s2;
          s1 = "Java";
          s2 = s1; // переменная ссылается на ту же строку
          System.out.println("сравнение ссылок "
             + (s1 == s2)); // результат true

          // создание нового объекта добавлением символа
          s1 += '2';
          // s1-="a"; //ошибка, вычитать строки нельзя
          // создание нового объекта копированием
          s2 = new String(s1);
          System.out.println("сравнение ссылок "
             + (s1 == s2)); // результат false

          System.out.println("сравнение значений "
             + s1.equals(s2)); // результат true
     }
}
```

Консоль

Взаимодействие с консолью с помощью потока **System.in** представляет собой один из простейших способов передачи информации в приложение. В следующем примере рассматривается ввод информации в виде символа из потока ввода, связанного с консолью, и последующего вывода на консоль символа и его числового кода.

```
// пример # 8 : чтение символа из потока System.in : DemoSystemIn.java
package chapt01;

public class ReadCharRunner {

  public static void main(String[] args) {
     int x;
     try {
        x = System.in.read();
        char c = (char)x;
        System.out.println("Код символа: " + c + " =" + x);
     } catch (java.io.IOException e) {
        e.printStackTrace();
     }
   }
}
```

Обработка исключительной ситуации **IOException**, которая возникает в операциях ввода/вывода и в любых других взаимодействиях с внешними устройствами, осуществляется в методе **main()** с помощью реализации блока **try-catch**.

Ввод блока информации осуществляется с помощью чтения строки из консоли. Далее строка может быть использована в исходном виде или преобразована к требуемому виду.

```
// пример # 9 : чтение строки из консоли : ReadCharRunner.java
package chapt01;
import java.io.*;//подключение пакета классов

public class ReadCharRunner {

      public static void main(String[] args) {
      /* байтовый поток ввода System.in передается конструктору потока
            чтения при создании объекта класса InputStreamReader  */
            InputStreamReader is =
                  new InputStreamReader(System.in);
      /* производится буферизация данных, исключающая необходимость
       обращения к источнику данных при выполнении операции чтения */
            BufferedReader bis = new BufferedReader(is);
            try {
                  System.out.println(
                  "Введите Ваше имя и нажмите <Enter>:");
      /* чтение строки из буфера; метод readLine() требует обработки
            возможной ошибки при вводе из консоли в блоке try */
                  String name = bis.readLine();
                  System.out.println("Привет, " + name);
            } catch (IOException e) {
                  System.err.print("ошибка ввода " + e);
            }
      }
}
```

В результате запуска приложения будет выведено, например, следующее:

```
Введите Ваше имя и нажмите <Enter>:
Остап
Привет, Остап
```

Позже будут рассмотрены более удобные способы извлечения информации из потока ввода, в качестве которого может фигурировать не только консоль, но и дисковый файл, сокетное соединение и пр.

Кроме того, в шестой версии языка существует возможность поддержать национальный шрифт с помощью метода **printf()** определенного для класса **Console**.

```
/* пример # 10 : использование метода printf() класса Console: PrintDeutsch.java */
public class PrintDeutsch {
      public static void main(String[] args) {
            String str = "über";
```

```
            System.out.println(str);
            Console con = System.console();
            con.printf("%s", str);
      }
}
```

В результате будет выведено:

□**ber**
über

Простой апплет

Одной из целей создания языка Java было создание апплетов – небольших программ, запускаемых Web-браузером. Поскольку апплеты должны быть безопасными, они ограничены в своих возможностях, хотя остаются мощным инструментом поддержки Web-программирования на стороне клиента.

```
// пример # 11 : простой апплет: FirstApplet.java
import java.awt.Graphics;
import java.util.Calendar;

public class FirstApplet extends javax.swing.JApplet {
      private Calendar calendar;

      public void init() {
            calendar = Calendar.getInstance();
            setSize(250,80);
      }
      public void paint(Graphics g) {
            g.drawString("Апплет запущен:", 20, 15);
            g.drawString(
                  calendar.getTime().toString(), 20, 35);
      }
}
```

Для вывода текущего времени и даты в этом примере был использован объект **Calendar** из пакета **java.util**. Метод **toString()** используется для преобразования информации, содержащейся в объекте, в строку для последующего вывода в апплет с помощью метода **drawString()**. Цифровые параметры этого метода обозначают горизонтальную и вертикальную координаты начала рисования строки, считая от левого верхнего угла апплета.

Апплету не нужен метод **main()** – код его запуска помещается в метод **init()** или **paint()**. Для запуска апплета нужно поместить ссылку на его класс в HTML-документ и просмотреть этот документ Web-браузером, поддерживающим Java. При этом можно обойтись очень простым фрагментом (тегом) **<applet>** в HTML-документе **view.html**:

```
<html><body>
<applet code= FirstApplet.class width=300 height=300>
</applet></body></html>
```