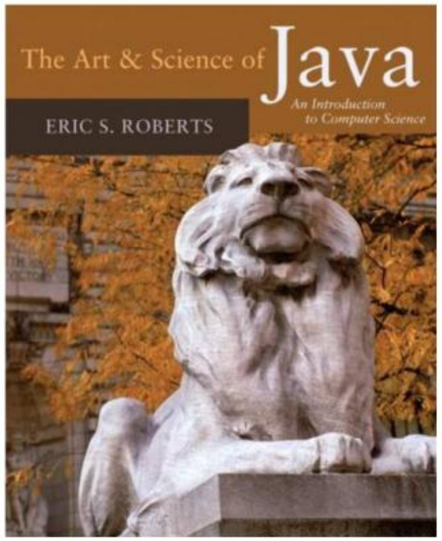
# ACM Java Tutorial

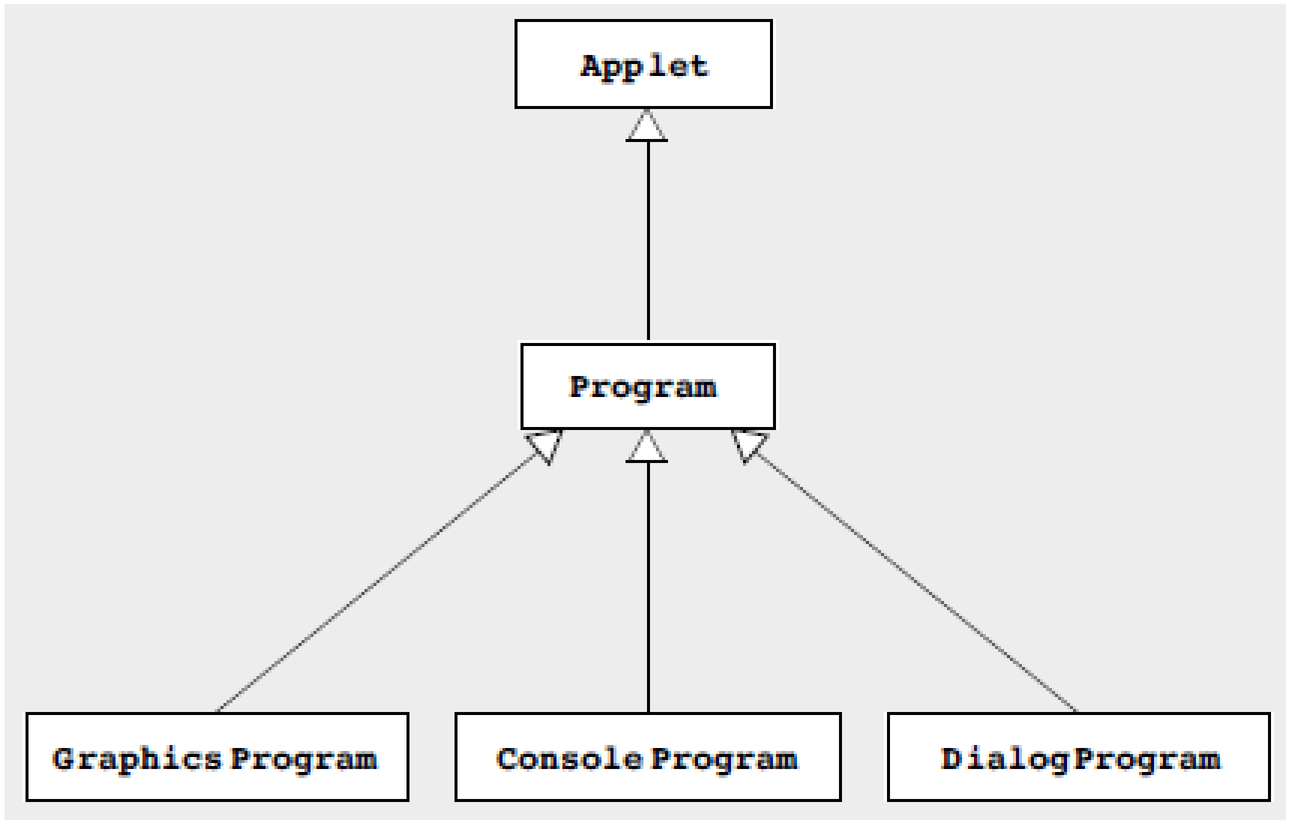
## ACM Library Resources

- Book: Free preliminary draft available

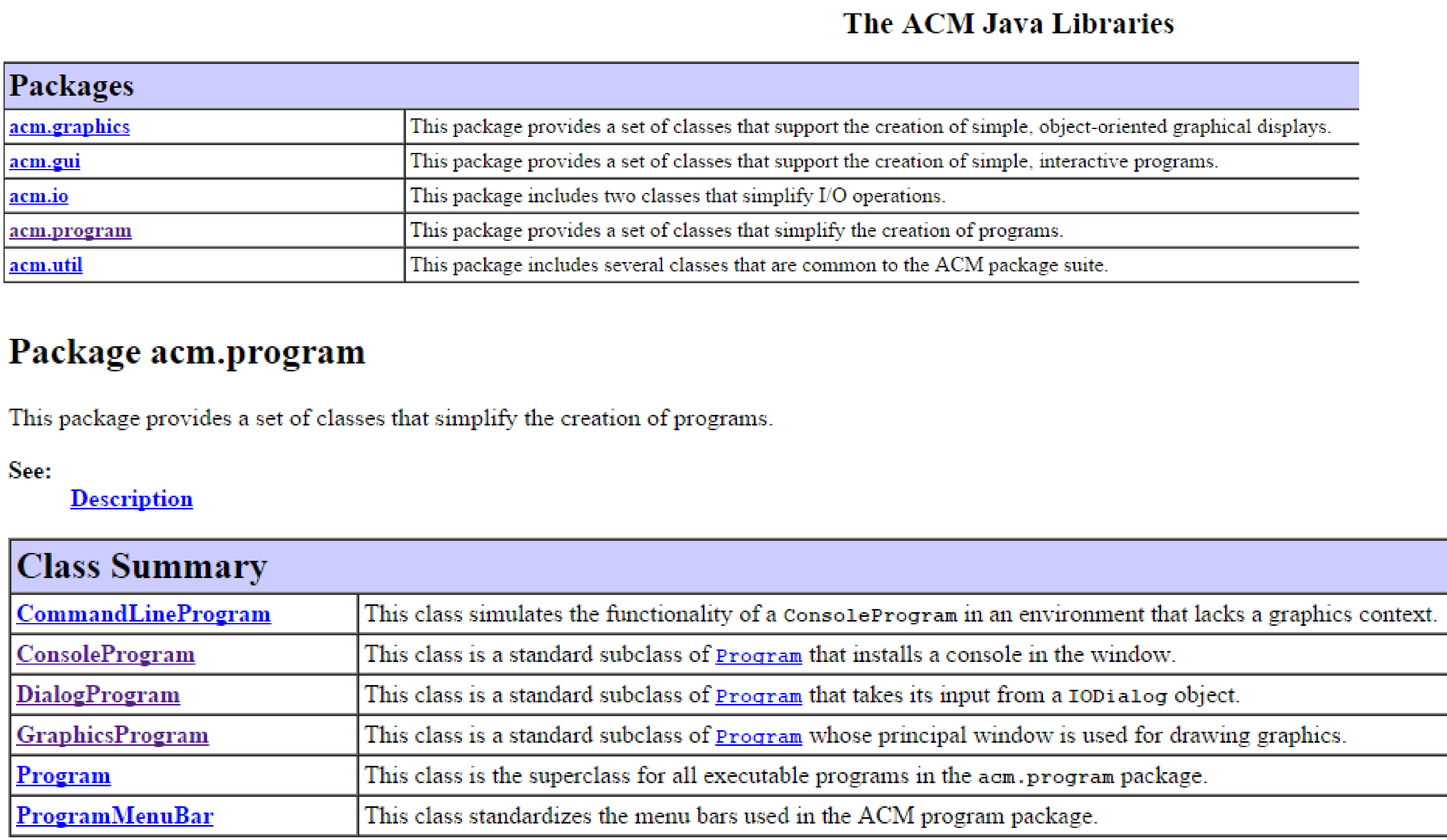

The Art & Science of Java
An Introduction to Computer Science
ERIC S. ROBERTS

- ACM Tutorial: Link
- ACM Demos: Link
- ACM Java API : Link

# The Program Class Hierarchy in ACM package



# ACM Library API



### The ACM Java Libraries

| Packages | |
|---|---|
| acm.graphics | This package provides a set of classes that support the creation of simple, object-oriented graphical displays. |
| acm.gui | This package provides a set of classes that support the creation of simple, interactive programs. |
| acm.io | This package includes two classes that simplify I/O operations. |
| acm.program | This package provides a set of classes that simplify the creation of programs. |
| acm.util | This package includes several classes that are common to the ACM package suite. |

## Package acm.program

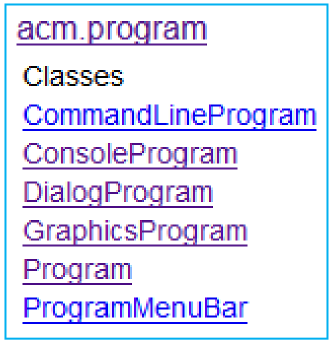
This package provides a set of classes that simplify the creation of programs.

**See:**
Description

| Class Summary | |
|---|---|
| CommandLineProgram | This class simulates the functionality of a `ConsoleProgram` in an environment that lacks a graphics context. |
| ConsoleProgram | This class is a standard subclass of `Program` that installs a console in the window. |
| DialogProgram | This class is a standard subclass of `Program` that takes its input from a `IODialog` object. |
| GraphicsProgram | This class is a standard subclass of `Program` whose principal window is used for drawing graphics. |
| Program | This class is the superclass for all executable programs in the `acm.program` package. |
| ProgramMenuBar | This class standardizes the menu bars used in the ACM program package. |

# Importing ACM Library classses

- Classes in ACM Java library are grouped into packages
- E.g. acm.programs package contains 6 classes:



acm.program

Classes

CommandLineProgram
ConsoleProgram
DialogProgram
GraphicsProgram
Program
ProgramMenuBar

- To import a specific class into your program, put an import statement at the beginning of the file
  - E.g. `import acm.program.DialogProgram;`
- To import all the classes contained in a particular package, use *:
  - E.g. `import acm.program.*;`

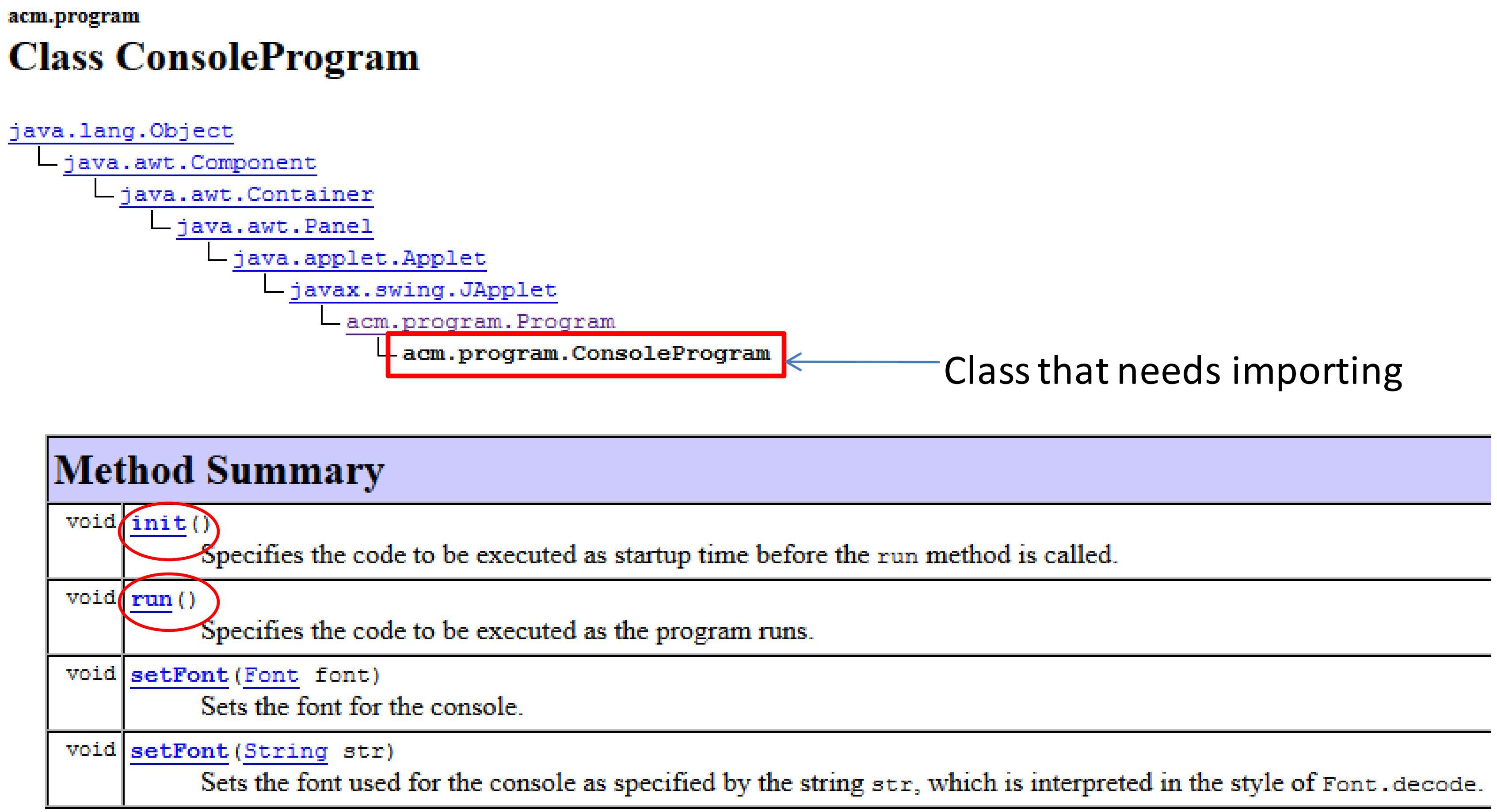
# ConsoleProgram

- Program that installs a console in the window.

- You need to import **acm.program.ConsoleProgram**

```
import acm.program.ConsoleProgram
```

- What are the methods avaiable in a ConsoleProgram?
  - Lookup ACM Java API documentation
    - http://cs.stanford.edu/people/eroberts/jtf/javadoc/student/index.html
    - **SELECT : acm.program> ConsoleProgram**



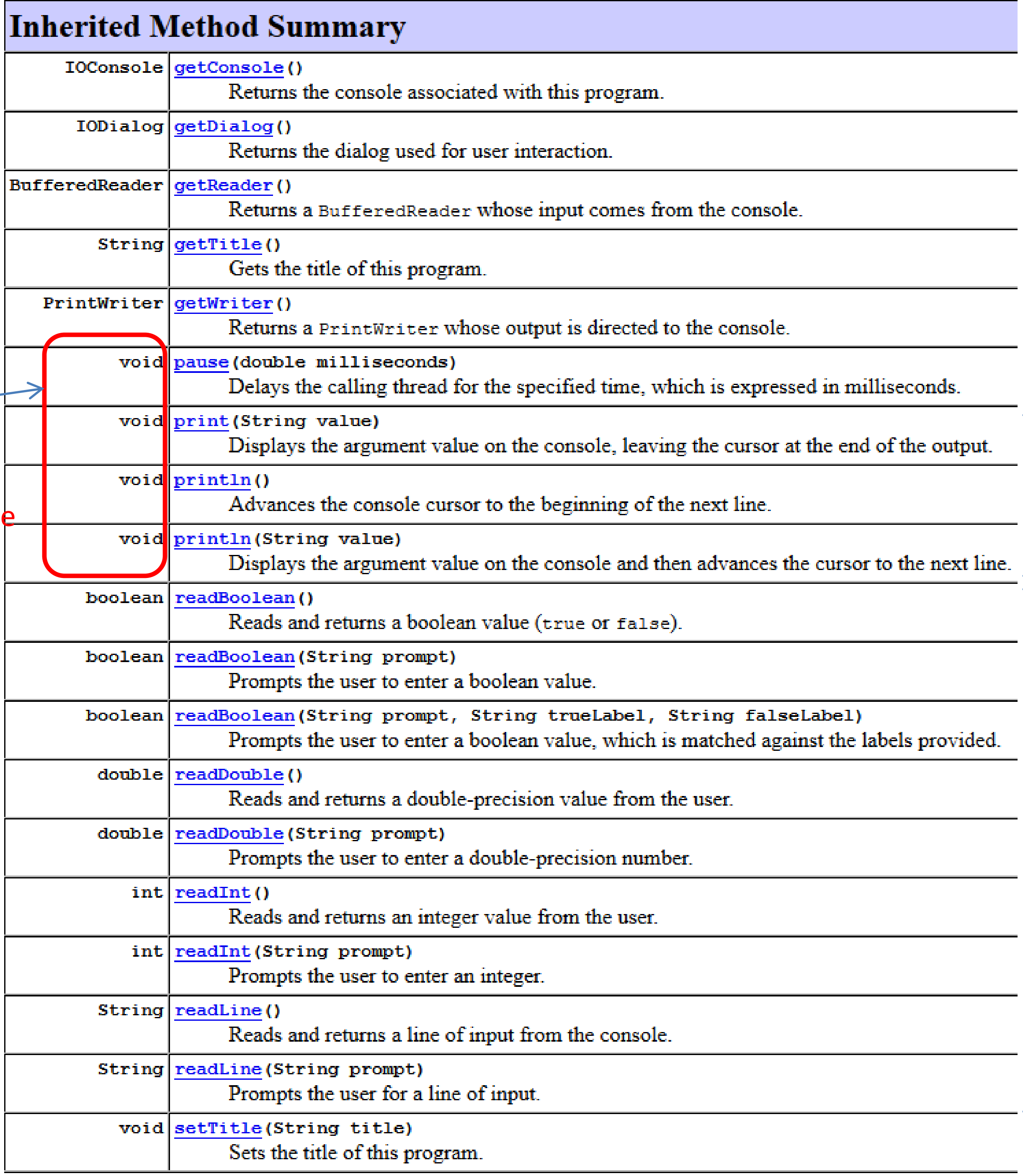
acm.program

**Class ConsoleProgram**

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Panel
              └java.applet.Applet
                  └javax.swing.JApplet
                      └acm.program.Program
                          └acm.program.ConsoleProgram
```

Class that needs importing

| Method Summary | |
|---|---|
| void | `init()` Specifies the code to be executed as startup time before the `run` method is called. |
| void | `run()` Specifies the code to be executed as the program runs. |
| void | `setFont(Font font)` Sets the font for the console. |
| void | `setFont(String str)` Sets the font used for the console as specified by the string `str`, which is interpreted in the style of `Font.decode`. |



## Inherited Method Summary

| | |
|---|---|
| `IOConsole` | `getConsole()`<br>Returns the console associated with this program. |
| `IODialog` | `getDialog()`<br>Returns the dialog used for user interaction. |
| `BufferedReader` | `getReader()`<br>Returns a `BufferedReader` whose input comes from the console. |
| `String` | `getTitle()`<br>Gets the title of this program. |
| `PrintWriter` | `getWriter()`<br>Returns a `PrintWriter` whose output is directed to the console. |
| `void` | `pause(double milliseconds)`<br>Delays the calling thread for the specified time, which is expressed in milliseconds. |
| `void` | `print(String value)`<br>Displays the argument value on the console, leaving the cursor at the end of the output. |
| `void` | `println()`<br>Advances the console cursor to the beginning of the next line. |
| `void` | `println(String value)`<br>Displays the argument value on the console and then advances the cursor to the next line. |
| `boolean` | `readBoolean()`<br>Reads and returns a boolean value (`true` or `false`). |
| `boolean` | `readBoolean(String prompt)`<br>Prompts the user to enter a boolean value. |
| `boolean` | `readBoolean(String prompt, String trueLabel, String falseLabel)`<br>Prompts the user to enter a boolean value, which is matched against the labels provided. |
| `double` | `readDouble()`<br>Reads and returns a double-precision value from the user. |
| `double` | `readDouble(String prompt)`<br>Prompts the user to enter a double-precision number. |
| `int` | `readInt()`<br>Reads and returns an integer value from the user. |
| `int` | `readInt(String prompt)`<br>Prompts the user to enter an integer. |
| `String` | `readLine()`<br>Reads and returns a line of input from the console. |
| `String` | `readLine(String prompt)`<br>Prompts the user for a line of input. |
| `void` | `setTitle(String title)`<br>Sets the title of this program. |

Void :
Method
does not
return a value

Print to console

Read user input
From console

# ConsolProgram : Syntax

Importing ConsoleProgram class

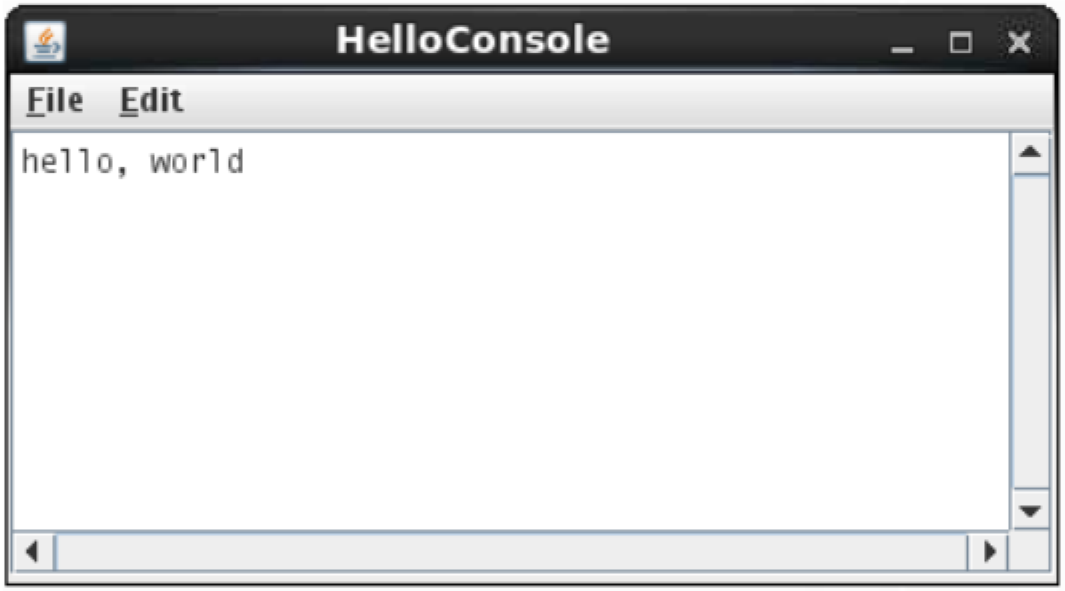
```
import acm.program.ConsoleProgram;

public class YourClassName extends ConsoleProgram
{
    public void run()
    {
            //your code here
    }

    public static void main(String[] args)
    {
            new YourClassName ().start(args);
    }
}
```

# ConsolProgram : Example1

- Example: prints “hello, world” to console

```
import acm.program.ConsoleProgram;

public class HelloConsole extends ConsoleProgram
{
          public void run()
          {
                println("hello, world");
          }

   public static void main(String[] args)
   {
                new HelloConsole ().start(args);
   }
}
```

## ConsoleProgram - Output



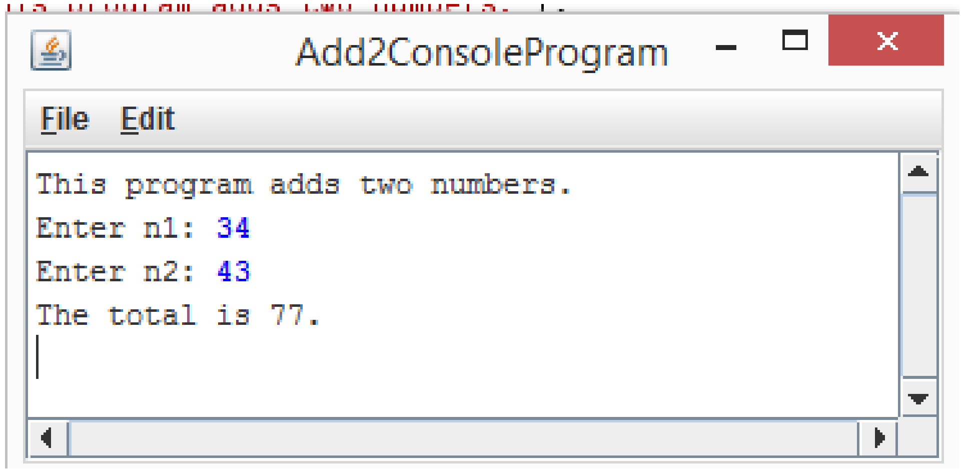
```
hello, world
```

# ConsoleProgram: Example2

- Program prompts the user for two numbers and prints  the total:

```
import acm.program.ConsoleProgram;

public class Add2Program extends ConsoleProgram
{
      public void run() {
            println("This program adds two numbers.");
            int n1 = readInt("Enter n1: ");
            int n2 = readInt("Enter n2: ");
            int total = n1 + n2;
            println("The total is " + total + ".");
      }

      public static void main(String[] args)
      {
            new Add2Program().start(args);
      }
}
```

# Example2: Output



```
This program adds two numbers.
Enter n1: 34
Enter n2: 43
The total is 77.
```

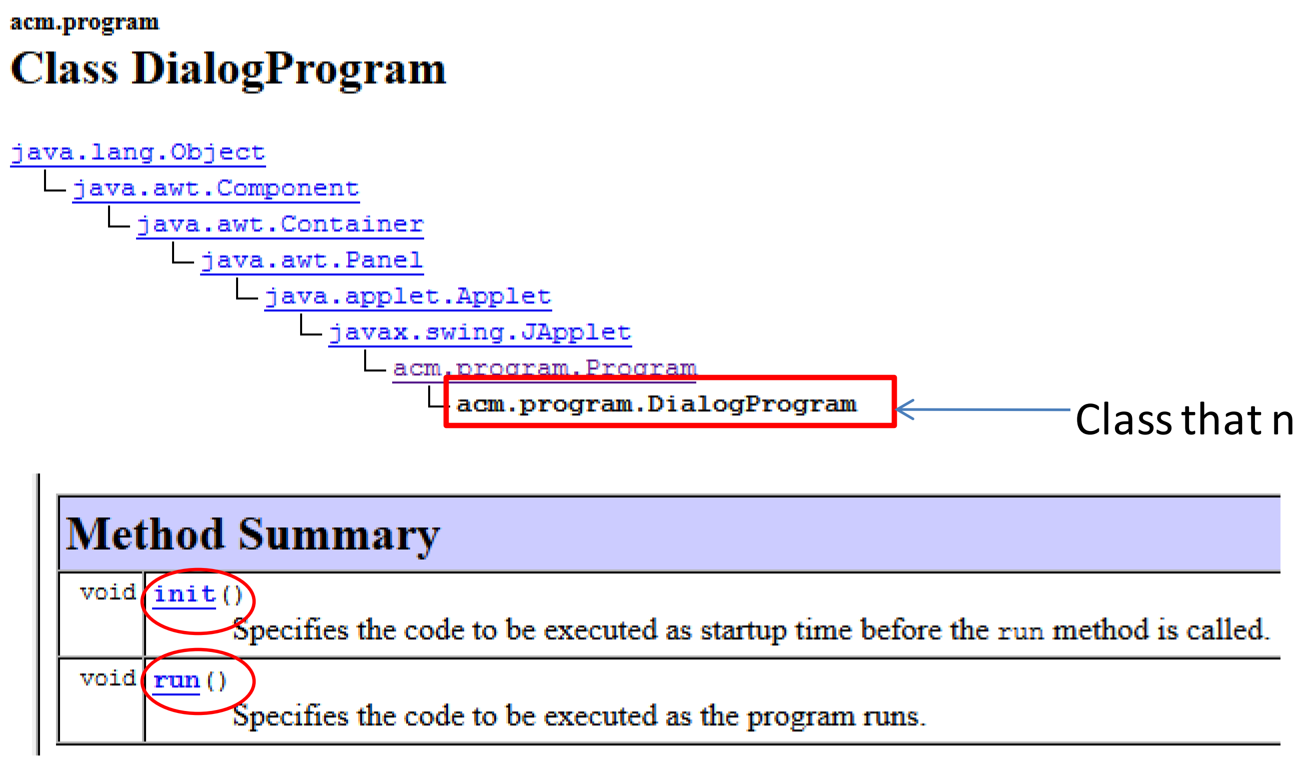
# DialogProgram

- Program that takes its input from a IODialog object

- You need to import **acm.program.DialogProgram**

```
import acm.program.DialogProgram
```

- What are the methods avaiable in a DialogProgram?
  - Lookup ACM Java API documentation
    - http://cs.stanford.edu/people/eroberts/jtf/javadoc/student/index.html
    - **SELECT : acm.program> DialogProgram**



acm.program

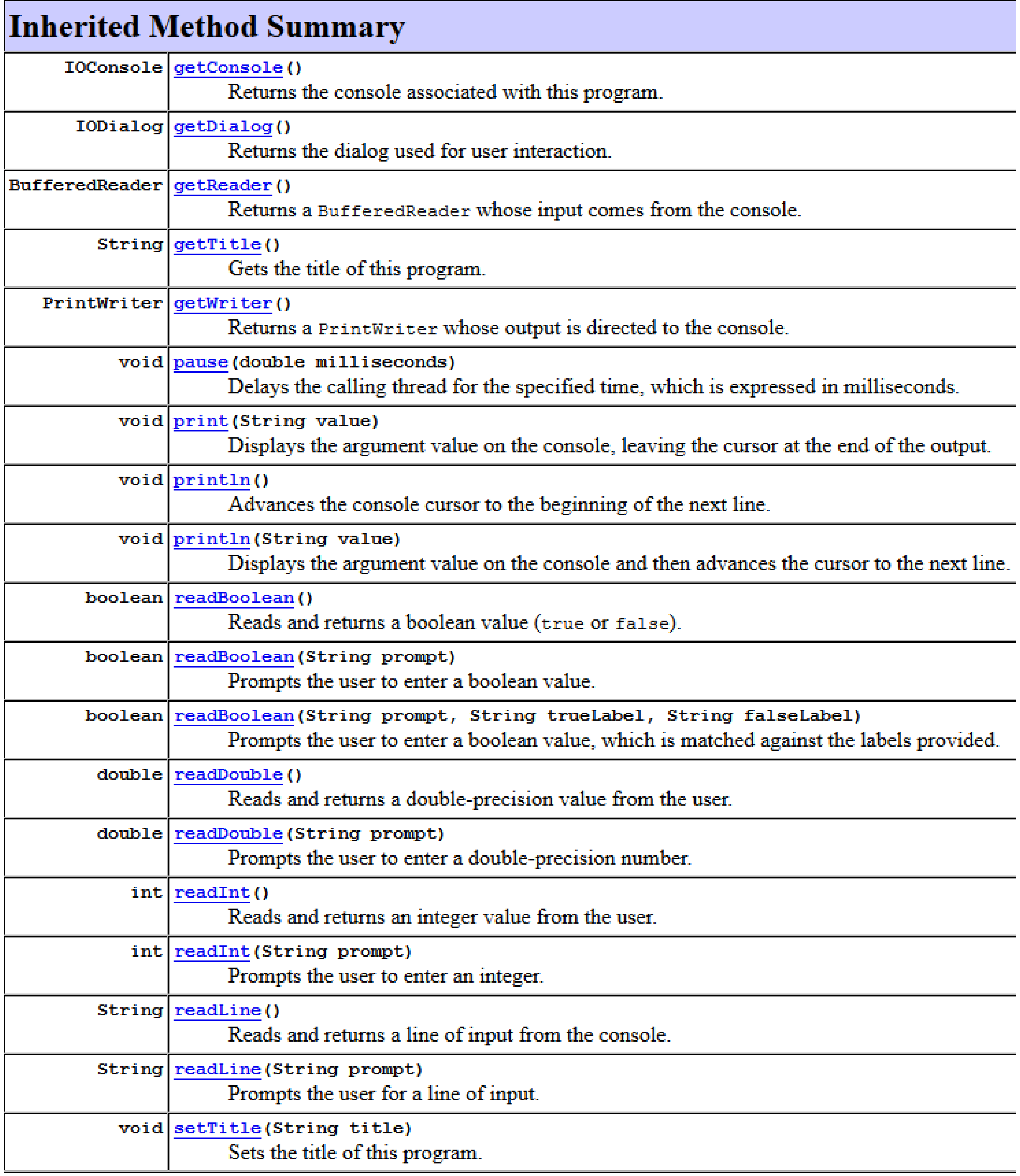
**Class DialogProgram**

```
java.lang.Object
  └ java.awt.Component
      └ java.awt.Container
          └ java.awt.Panel
              └ java.applet.Applet
                  └ javax.swing.JApplet
                      └ acm.program.Program
                          └ acm.program.DialogProgram
```

Class that needs importing

**Method Summary**

| | |
|---|---|
| void | init() <br> Specifies the code to be executed as startup time before the `run` method is called. |
| void | run() <br> Specifies the code to be executed as the program runs. |



## Inherited Method Summary

| | |
|---|---|
| IOConsole | getConsole() <br> Returns the console associated with this program. |
| IODialog | getDialog() <br> Returns the dialog used for user interaction. |
| BufferedReader | getReader() <br> Returns a BufferedReader whose input comes from the console. |
| String | getTitle() <br> Gets the title of this program. |
| PrintWriter | getWriter() <br> Returns a PrintWriter whose output is directed to the console. |
| void | pause(double milliseconds) <br> Delays the calling thread for the specified time, which is expressed in milliseconds. |
| void | print(String value) <br> Displays the argument value on the console, leaving the cursor at the end of the output. |
| void | println() <br> Advances the console cursor to the beginning of the next line. |
| void | println(String value) <br> Displays the argument value on the console and then advances the cursor to the next line. |
| boolean | readBoolean() <br> Reads and returns a boolean value (true or false). |
| boolean | readBoolean(String prompt) <br> Prompts the user to enter a boolean value. |
| boolean | readBoolean(String prompt, String trueLabel, String falseLabel) <br> Prompts the user to enter a boolean value, which is matched against the labels provided. |
| double | readDouble() <br> Reads and returns a double-precision value from the user. |
| double | readDouble(String prompt) <br> Prompts the user to enter a double-precision number. |
| int | readInt() <br> Reads and returns an integer value from the user. |
| int | readInt(String prompt) <br> Prompts the user to enter an integer. |
| String | readLine() <br> Reads and returns a line of input from the console. |
| String | readLine(String prompt) <br> Prompts the user for a line of input. |
| void | setTitle(String title) <br> Sets the title of this program. |

Exactly same as
for ConsoleProgram

Print to console

Read user input
From console

# DialogProgram : Syntax

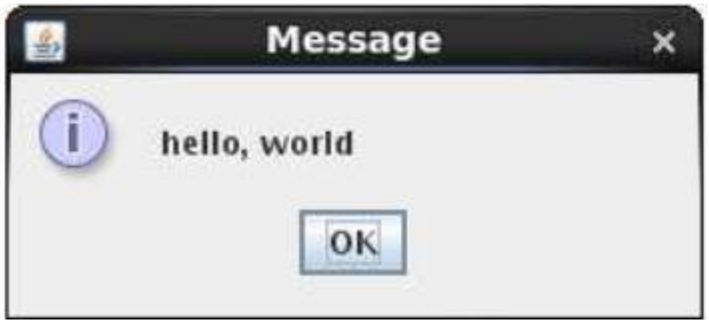
Same as Console program. Only difference is it extends DialogProgram

```
import acm.program.DialogProgram;

public class YourClassName extends DialogProgram
{
    public void run()
    {
          //your code here
    }

    public static void main(String[] args)
    {
          new YourClassName ().start(args);
    }
}
```

# DialogProgram : Example

```
import acm.program.DialogProgram;

public class HelloDialog extends DialogProgram
{
         public void run()
         {
               println("hello, world");
         }

   public static void main(String[] args)
   {
               new HelloDialog ().start(args);
   }
}
```

# DialogProgram - Output



```
hello, world
```

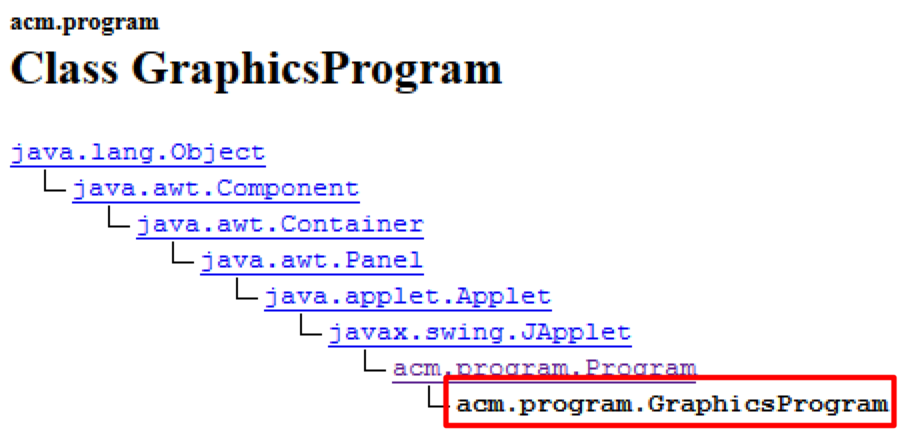
# GraphicsProgram

- Program that takes its input from a IODialog object

- You need to import **acm.program.GraphicsProgram**

```
import acm.program.GraphicsProgram
```

- What are the methods avaiable in a GraphicsProgram?
  - Lookup ACM Java API documentation
    - http://cs.stanford.edu/people/eroberts/jtf/javadoc/student/index.html
    - **SELECT : acm.program> GraphicsProgram**

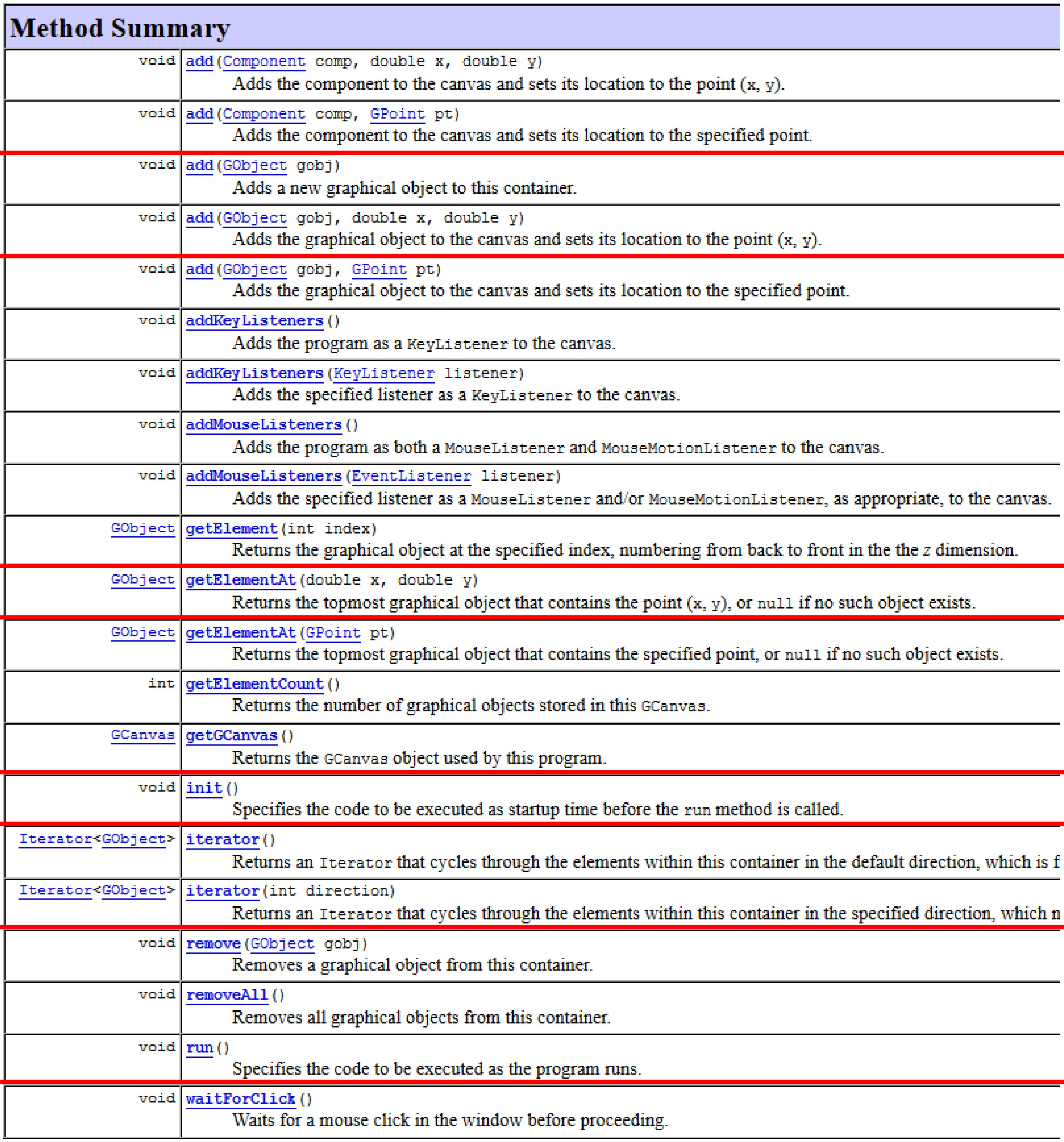


acm.program

**Class GraphicsProgram**

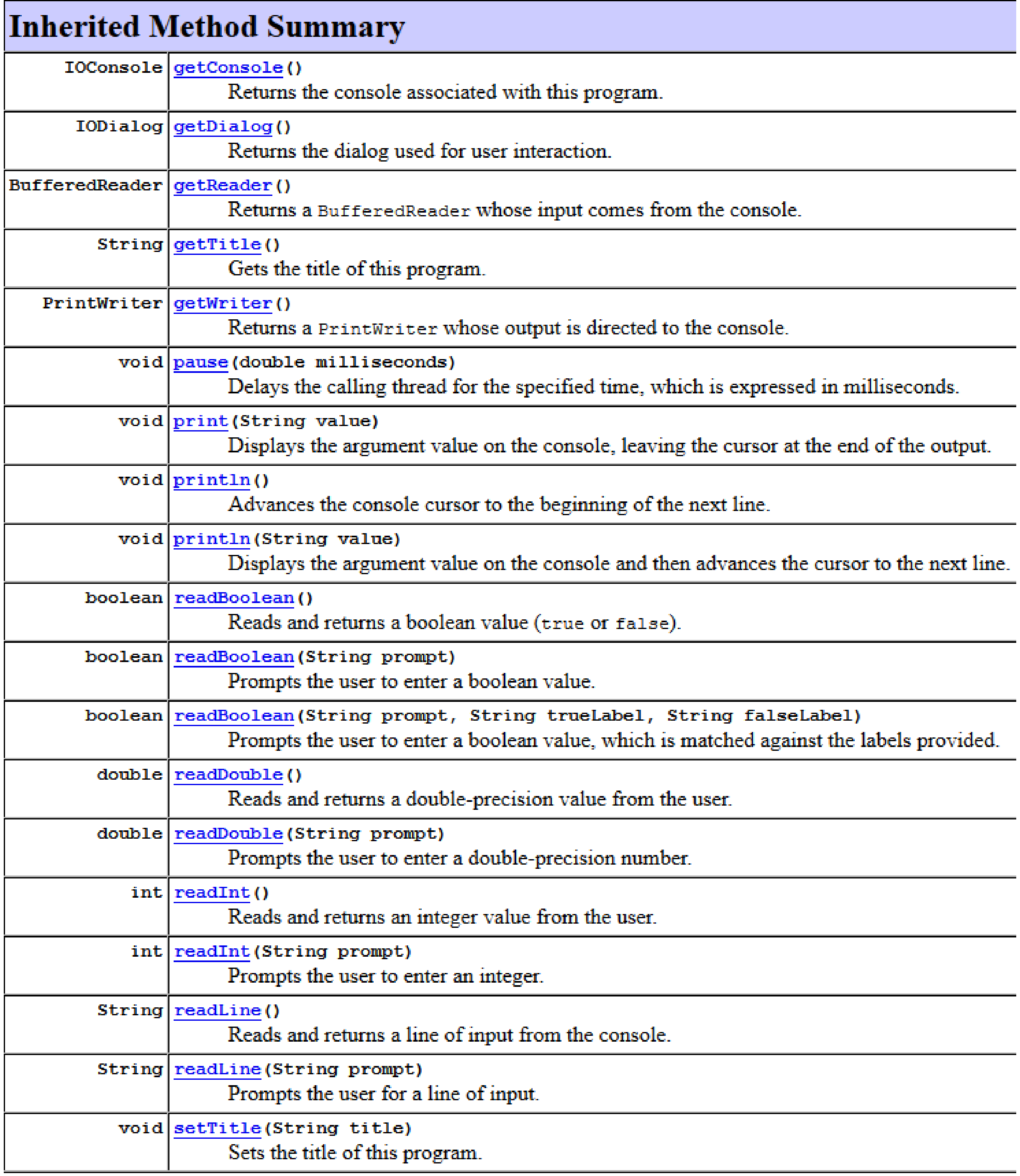
```
java.lang.Object
  └ java.awt.Component
      └ java.awt.Container
          └ java.awt.Panel
              └ java.applet.Applet
                  └ javax.swing.JApplet
                      └ acm.program.Program
                          └ acm.program.GraphicsProgram
```

Class that needs importing



## Method Summary

| | |
|---|---|
| `void` | `add(Component comp, double x, double y)`<br>Adds the component to the canvas and sets its location to the point (x, y). |
| `void` | `add(Component comp, GPoint pt)`<br>Adds the component to the canvas and sets its location to the specified point. |
| `void` | `add(GObject gobj)`<br>Adds a new graphical object to this container. |
| `void` | `add(GObject gobj, double x, double y)`<br>Adds the graphical object to the canvas and sets its location to the point (x, y). |
| `void` | `add(GObject gobj, GPoint pt)`<br>Adds the graphical object to the canvas and sets its location to the specified point. |
| `void` | `addKeyListeners()`<br>Adds the program as a `KeyListener` to the canvas. |
| `void` | `addKeyListeners(KeyListener listener)`<br>Adds the specified listener as a `KeyListener` to the canvas. |
| `void` | `addMouseListeners()`<br>Adds the program as both a `MouseListener` and `MouseMotionListener` to the canvas. |
| `void` | `addMouseListeners(EventListener listener)`<br>Adds the specified listener as a `MouseListener` and/or `MouseMotionListener`, as appropriate, to the canvas. |
| `GObject` | `getElement(int index)`<br>Returns the graphical object at the specified index, numbering from back to front in the the *z* dimension. |
| `GObject` | `getElementAt(double x, double y)`<br>Returns the topmost graphical object that contains the point (x, y), or `null` if no such object exists. |
| `GObject` | `getElementAt(GPoint pt)`<br>Returns the topmost graphical object that contains the specified point, or `null` if no such object exists. |
| `int` | `getElementCount()`<br>Returns the number of graphical objects stored in this `GCanvas`. |
| `GCanvas` | `getGCanvas()`<br>Returns the `GCanvas` object used by this program. |
| `void` | `init()`<br>Specifies the code to be executed as startup time before the `run` method is called. |
| `Iterator<GObject>` | `iterator()`<br>Returns an `Iterator` that cycles through the elements within this container in the default direction, which is f |
| `Iterator<GObject>` | `iterator(int direction)`<br>Returns an `Iterator` that cycles through the elements within this container in the specified direction, which n |
| `void` | `remove(GObject gobj)`<br>Removes a graphical object from this container. |
| `void` | `removeAll()`<br>Removes all graphical objects from this container. |
| `void` | `run()`<br>Specifies the code to be executed as the program runs. |
| `void` | `waitForClick()`<br>Waits for a mouse click in the window before proceeding. |



## Inherited Method Summary

| | |
|---|---|
| `IOConsole` | `getConsole()`<br>Returns the console associated with this program. |
| `IODialog` | `getDialog()`<br>Returns the dialog used for user interaction. |
| `BufferedReader` | `getReader()`<br>Returns a `BufferedReader` whose input comes from the console. |
| `String` | `getTitle()`<br>Gets the title of this program. |
| `PrintWriter` | `getWriter()`<br>Returns a `PrintWriter` whose output is directed to the console. |
| `void` | `pause(double milliseconds)`<br>Delays the calling thread for the specified time, which is expressed in milliseconds. |
| `void` | `print(String value)`<br>Displays the argument value on the console, leaving the cursor at the end of the output. |
| `void` | `println()`<br>Advances the console cursor to the beginning of the next line. |
| `void` | `println(String value)`<br>Displays the argument value on the console and then advances the cursor to the next line. |
| `boolean` | `readBoolean()`<br>Reads and returns a boolean value (`true` or `false`). |
| `boolean` | `readBoolean(String prompt)`<br>Prompts the user to enter a boolean value. |
| `boolean` | `readBoolean(String prompt, String trueLabel, String falseLabel)`<br>Prompts the user to enter a boolean value, which is matched against the labels provided. |
| `double` | `readDouble()`<br>Reads and returns a double-precision value from the user. |
| `double` | `readDouble(String prompt)`<br>Prompts the user to enter a double-precision number. |
| `int` | `readInt()`<br>Reads and returns an integer value from the user. |
| `int` | `readInt(String prompt)`<br>Prompts the user to enter an integer. |
| `String` | `readLine()`<br>Reads and returns a line of input from the console. |
| `String` | `readLine(String prompt)`<br>Prompts the user for a line of input. |
| `void` | `setTitle(String title)`<br>Sets the title of this program. |

Exactly same as
for ConsoleProgram

Print to console

Read user input
From console

## GraphicsProgram : Syntax

```
import acm.graphics.*;
import acm.program.GraphicsProgram;

public class YourClassName extends GraphicsProgram
{
  public void run()
  {
    //your code here
  }

  public static void main(String[] args) {
    new YourClassName().start(args);
  }
}
```

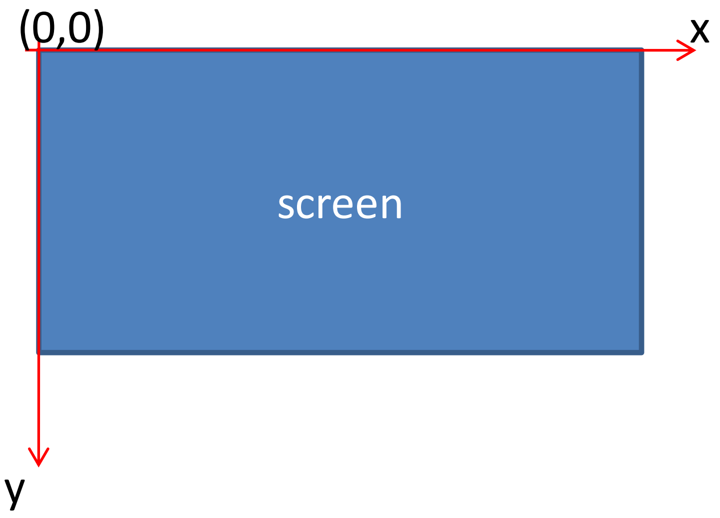
## GraphicsProgram : Example

```
import acm.graphics.GLabel;
import acm.program.GraphicsProgram;

public class HelloGraphics extends GraphicsProgram
{
  public void run()
  {
    GLabel label = new GLabel("hello, world");
    label.setFont("SansSerif-100");
    double x = (getWidth() - label.getWidth()) / 2;
    double y = (getHeight() + label.getAscent()) / 2;
    add(label, x, y);
  }

  public static void main(String[] args)
  {
    new HelloGraphics().start(args);
  }
}
```

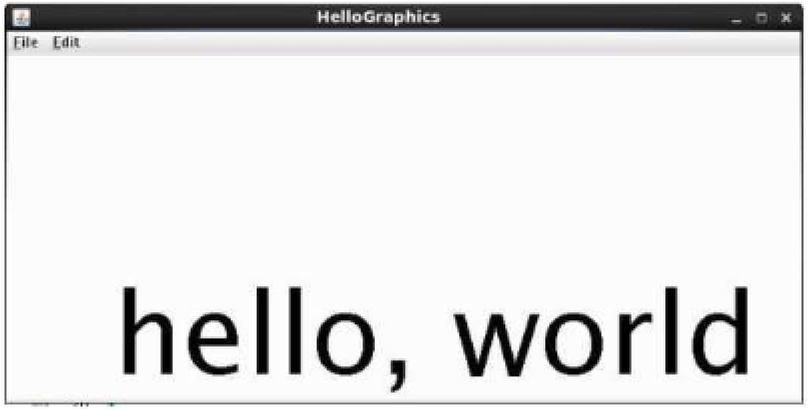
# GraphicsProgram : Example

```
import acm.graphics.GLabel;
import acm.program.GraphicsProgram;

public class HelloGraphics extends GraphicsProgram
{
  public void run()
  {
    //create a GLabel object with message text
    GLabel label = new GLabel("hello, world");
    //Give label object text a large font
    label.setFont("SansSerif-100");
    //next 3 lines add label so it is centered in the window
    double x = (getWidth() - label.getWidth()) / 2;
    double y = (getHeight() + label.getAscent()) / 2;
    add(label, x, y);
  }

  public static void main(String[] args) {
    new HelloGraphics().start(args);
  }
}
```

- Coordinate System:
  - Given in pixels
  - Graphics objects can be placed in specific coordinates
    - E.g. `add(label, x, y);`

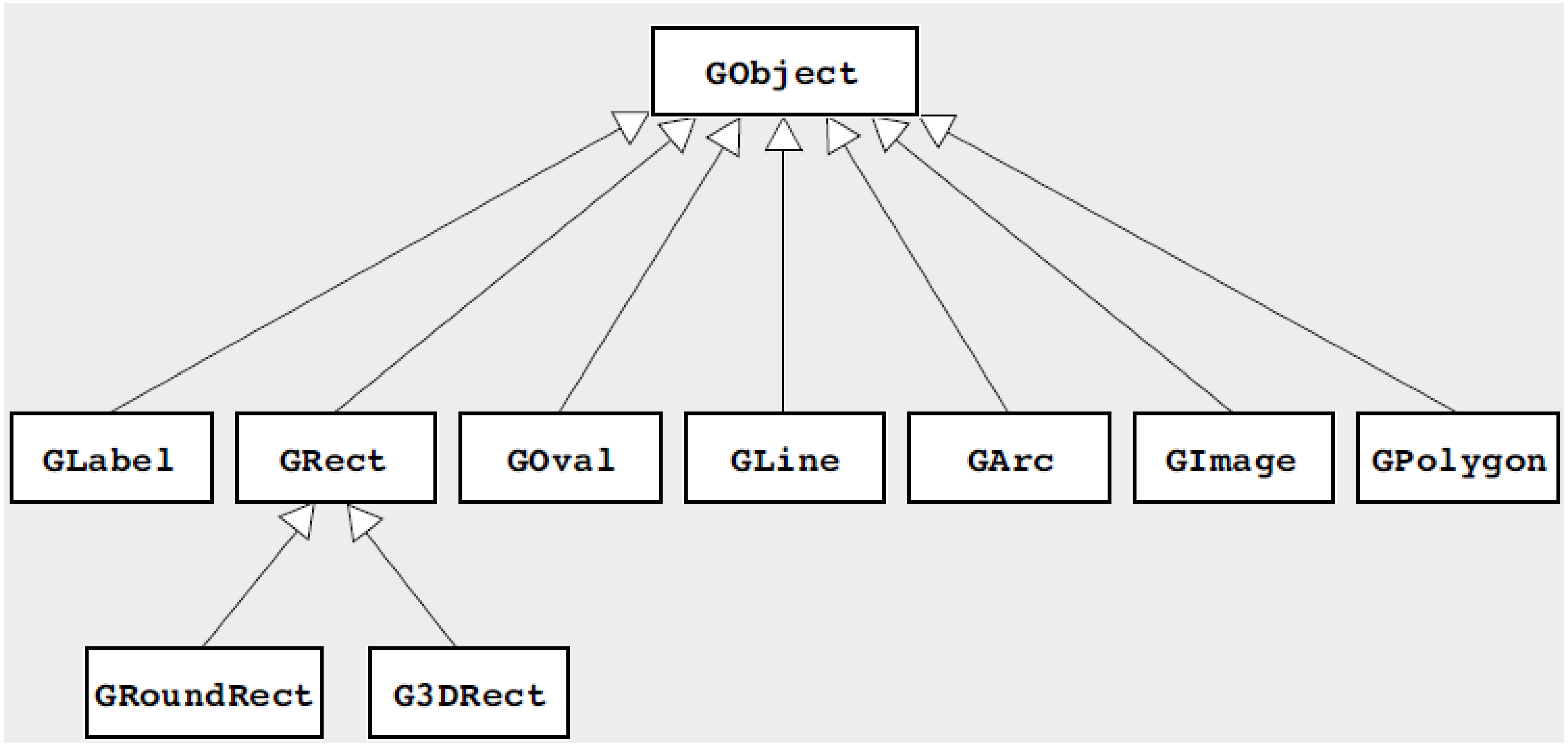


# GraphicsProgram - Output



```
hello, world
```

# acm.graphics package



# How to create a object

- Similar to creating GLabel object
- E.g.

```
GLabel labelA = new GLabel("Hello World");
```

- Syntax:

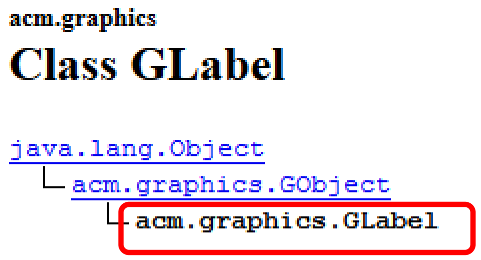
```
ClassName objectName = new ClassName([arguments]);
```

Look at constructor list in ACM Java API to find arguments

[] denote optional arguments

# GLabel

- Importing: `import acm.graphics.GLabel;`



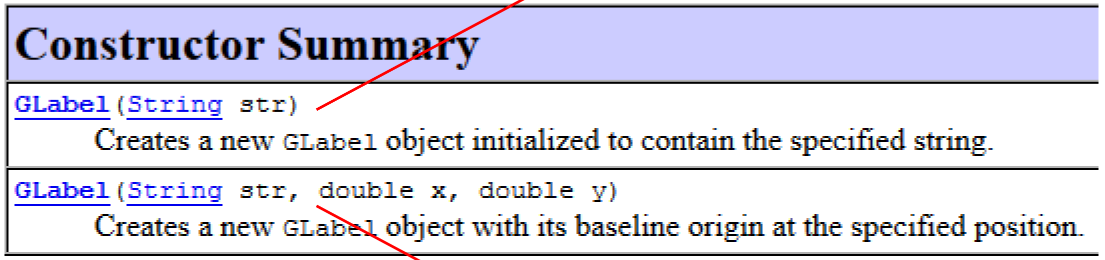
acm.graphics

**Class GLabel**

```
java.lang.Object
  └ acm.graphics.GObject
      └ acm.graphics.GLabel
```

- Creating objects:

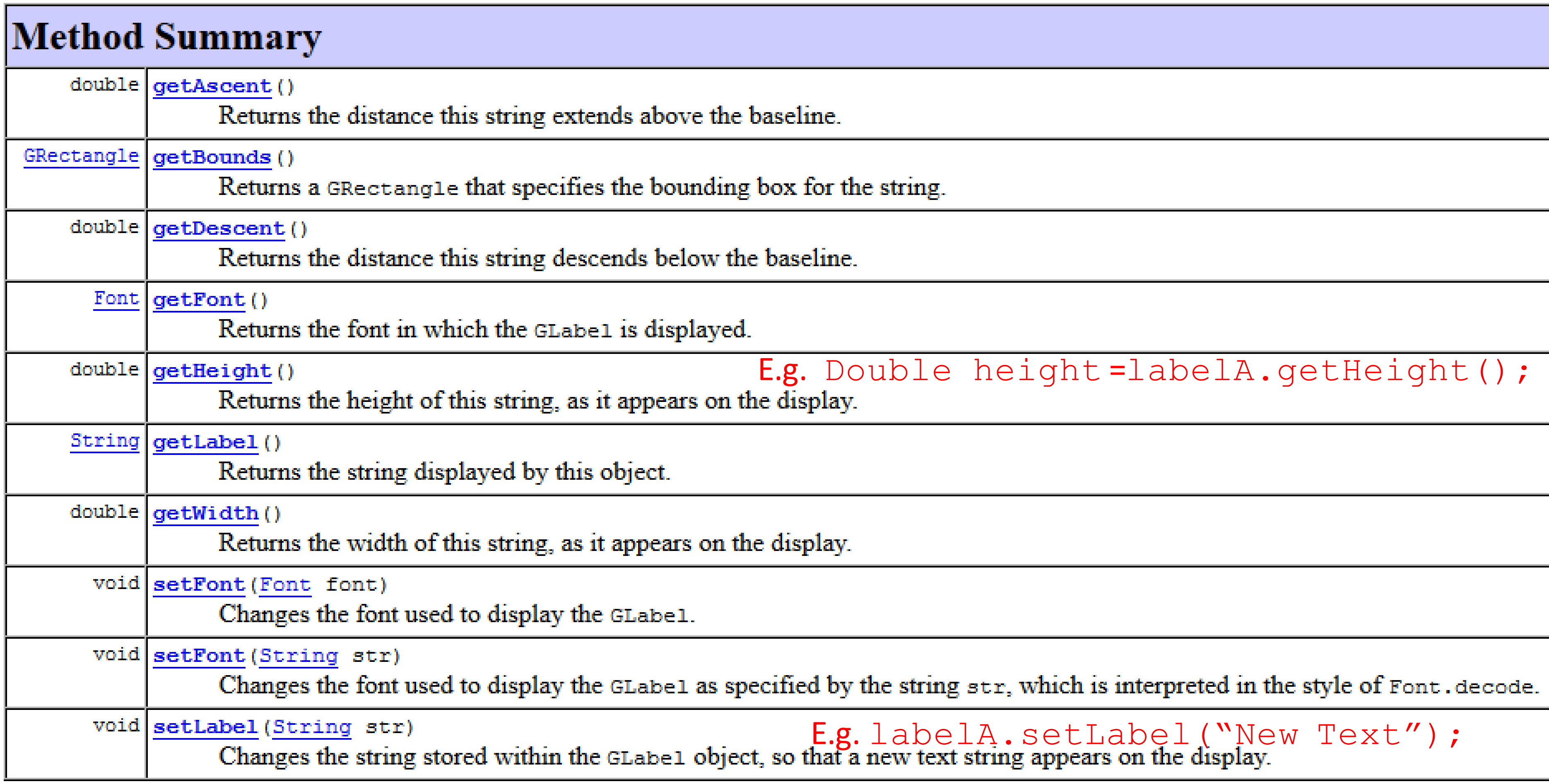
E.g. `GLabel labelA = new Glabel("Hello");`



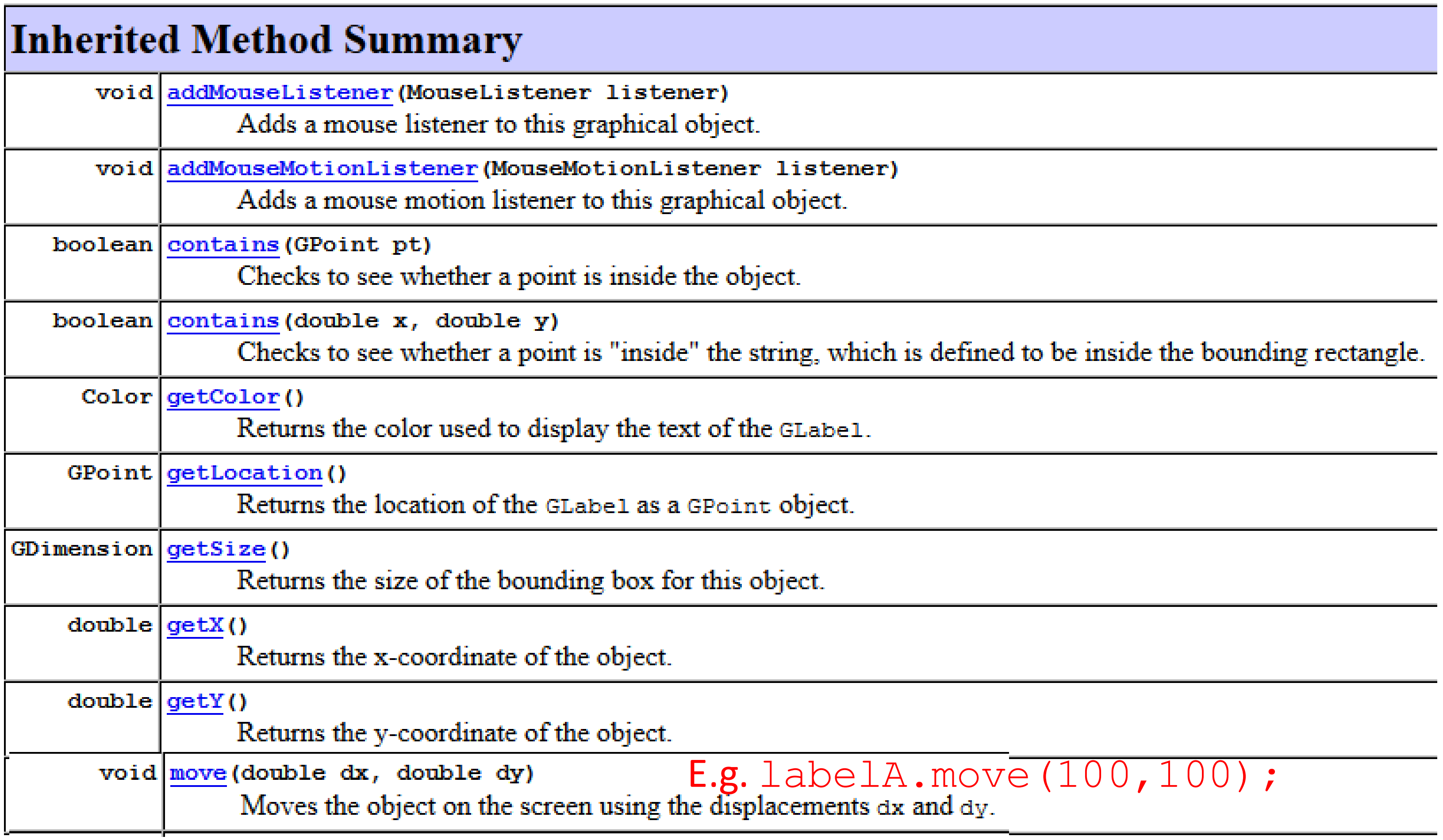
| **Constructor Summary** |
|---|
| `GLabel(String str)`<br>Creates a new `GLabel` object initialized to contain the specified string. |
| `GLabel(String str, double x, double y)`<br>Creates a new `GLabel` object with its baseline origin at the specified position. |

E.g. `GLabel labelB = new Glabel("Hello", 20, 30);`

- Call method syntax:
  - [returnValue =] objectName.methodName([arguments]);



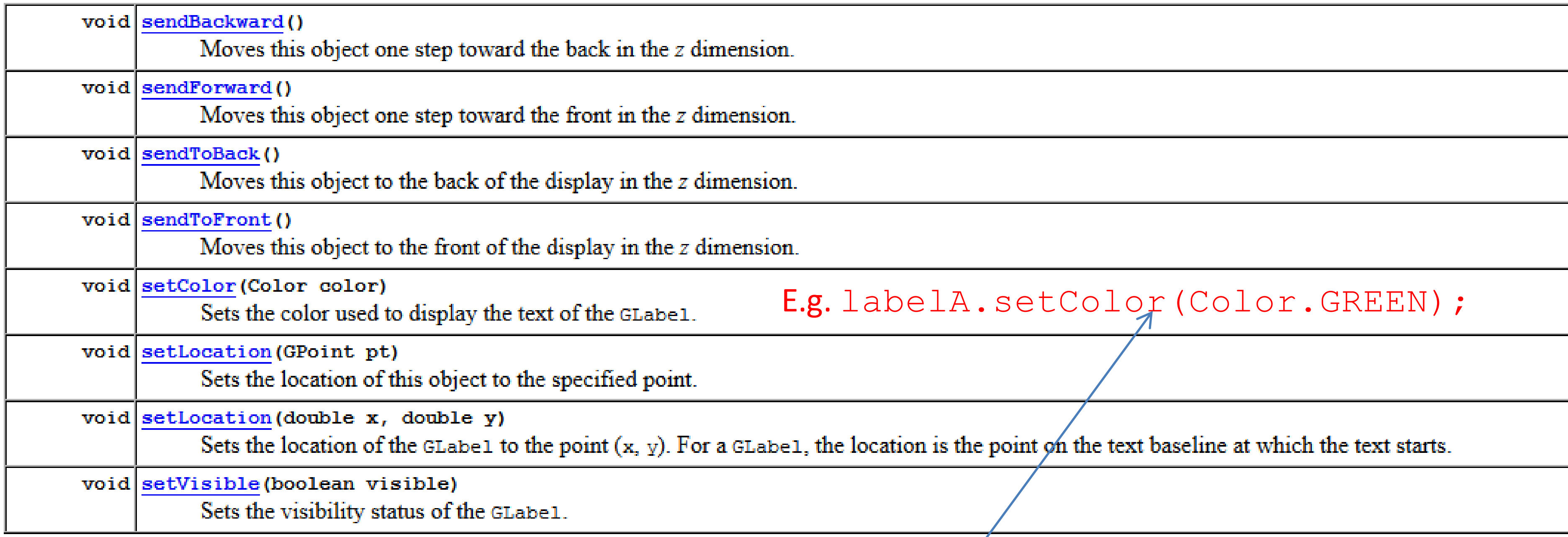
| Method Summary | |
|---|---|
| double | getAscent() <br> Returns the distance this string extends above the baseline. |
| GRectangle | getBounds() <br> Returns a GRectangle that specifies the bounding box for the string. |
| double | getDescent() <br> Returns the distance this string descends below the baseline. |
| Font | getFont() <br> Returns the font in which the GLabel is displayed. |
| double | getHeight() E.g. Double height =labelA.getHeight(); <br> Returns the height of this string, as it appears on the display. |
| String | getLabel() <br> Returns the string displayed by this object. |
| double | getWidth() <br> Returns the width of this string, as it appears on the display. |
| void | setFont(Font font) <br> Changes the font used to display the GLabel. |
| void | setFont(String str) <br> Changes the font used to display the GLabel as specified by the string str, which is interpreted in the style of Font.decode. |
| void | setLabel(String str) E.g. labelA.setLabel("New Text"); <br> Changes the string stored within the GLabel object, so that a new text string appears on the display. |



## Inherited Method Summary

| | |
|---|---|
| `void` | `addMouseListener(MouseListener listener)` Adds a mouse listener to this graphical object. |
| `void` | `addMouseMotionListener(MouseMotionListener listener)` Adds a mouse motion listener to this graphical object. |
| `boolean` | `contains(GPoint pt)` Checks to see whether a point is inside the object. |
| `boolean` | `contains(double x, double y)` Checks to see whether a point is "inside" the string, which is defined to be inside the bounding rectangle. |
| `Color` | `getColor()` Returns the color used to display the text of the `GLabel`. |
| `GPoint` | `getLocation()` Returns the location of the `GLabel` as a `GPoint` object. |
| `GDimension` | `getSize()` Returns the size of the bounding box for this object. |
| `double` | `getX()` Returns the x-coordinate of the object. |
| `double` | `getY()` Returns the y-coordinate of the object. |
| `void` | `move(double dx, double dy)` Moves the object on the screen using the displacements `dx` and `dy`. |

E.g. `labelA.move(100,100);`



| | |
|---|---|
| void | sendBackward()<br>Moves this object one step toward the back in the *z* dimension. |
| void | sendForward()<br>Moves this object one step toward the front in the *z* dimension. |
| void | sendToBack()<br>Moves this object to the back of the display in the *z* dimension. |
| void | sendToFront()<br>Moves this object to the front of the display in the *z* dimension. |
| void | setColor(Color color)<br>Sets the color used to display the text of the `GLabel`. |
| void | setLocation(GPoint pt)<br>Sets the location of this object to the specified point. |
| void | setLocation(double x, double y)<br>Sets the location of the `GLabel` to the point (*x*, *y*). For a `GLabel`, the location is the point on the text baseline at which the text starts. |
| void | setVisible(boolean visible)<br>Sets the visibility status of the `GLabel`. |

E.g. `labelA.setColor(Color.GREEN);`

Need to import java.awt.Color

Try changing font color to green in your HelloGraphics program!

# GLabel

- Creates a Label to place some text.
- Import: `import acm.graphics.GLabel`
- E.g.

```
GLabel label1= new GLabel(“text1”);
GLabel label2= new GLabel(“text2”, 10,20);
label2.setColor(Color.YELLOW);
boolean visible = label1.isVisible();
```

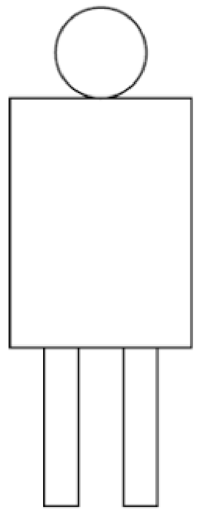
# GRect

- Creates a rectangle.
- Import: `import acm.graphics.GRect`
- E.g.

```
                                        GRect(double width, double height)
GRect rect1 = new GRect(10,15);
                                        GRect(double x, double y, double width, double height)
Grect rect2 = new Grect(0,0, 10, 15);
                                        setFillColor(Color color)
rect1.setFillColor(Color.green);
                               scale(double sf)
rect2.scale(0.5);
                               move(double dx, double dy)
rect2.move(10,10);
```

# Programming Question

- Write a GraphicsProgram subclass DrawRobot.java that generates the following picture of a robot. (Use HelloGraphics program as a template) Play with the coordinates until you get something that looks more or less right. Show your work to TA.



# Reading Exercise

- Read chapters 1,2 of ACM Java tutorial.
  - http://cs.stanford.edu/people/eroberts/jtf/tutorial/Tutorial.pdf

- Try all examples and understand.