How to run an Applet?

There are two ways to run an applet

1. By html file.
2. By appletViewer tool (for testing purpose).

Simple example of Applet by html file:

To execute the applet by html file, create an applet and compile it. After that create an html file and place the applet code in html file. Now click the html file.

```java
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{

public void paint(Graphics g){
g.drawString("welcome",150,150);
}

}
```

Note: class must be public because its object is created by Java Plugin software that resides on the browser.

myapplet.html

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

Simple example of Applet by appletviewer tool:

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. After that run it by: appletviewer First.java. Now Html file is not required but it is for testing purpose only.

```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{

public void paint(Graphics g){
g.drawString("welcome to applet",150,150);
}

}
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
```

To execute the applet by appletviewer tool, write in command prompt:

```
c:\>javac First.java
c:\>appletviewer First.java
```

Displaying Graphics in Applet

java.awt.Graphics class provides many methods for graphics programming.

Commonly used methods of Graphics class:

1. **public abstract void drawString(String str, int x, int y):** is used to draw the specified string.
2. **public void drawRect(int x, int y, int width, int height):** draws a rectangle with the specified width and height.
3. **public abstract void fillRect(int x, int y, int width, int height):** is used to fill rectangle with the default color and specified width and height.
4. **public abstract void drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height.

5. **public abstract void fillOval(int x, int y, int width, int height):** is used to fill oval with the default color and specified width and height.
6. **public abstract void drawLine(int x1, int y1, int x2, int y2):** is used to draw line between the points(x1, y1) and (x2, y2).
7. **public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer):** is used draw the specified image.
8. **public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used draw a circular or elliptical arc.
9. **public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used to fill a circular or elliptical arc.
10. **public abstract void setColor(Color c):** is used to set the graphics current color to the specified color.
11. **public abstract void setFont(Font font):** is used to set the graphics current font to the specified font.

Q44. Write a program in java to demonstrate use of Graphics in applet:

```java
import java.applet.Applet;
import java.awt.*;

public class GraphicsDemo extends Applet{

public void paint(Graphics g){
g.setColor(Color.red);
g.drawString("Welcome",50, 50);
g.drawLine(20,30,20,300);
g.drawRect(70,100,30,30);
g.fillRect(170,100,30,30);
g.drawOval(70,200,30,30);

g.setColor(Color.pink);
g.fillOval(170,200,30,30);
g.drawArc(90,150,30,30,30,270);
g.fillArc(270,150,30,30,0,180);

}
}
```

myapplet.html

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

Java ActionListener Interface

The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package. It has only one method: actionPerformed().

actionPerformed() method

The actionPerformed() method is invoked automatically whenever you click on the registered component.

```java
public abstract void actionPerformed(ActionEvent e);
```

How to write ActionListener

The common approach is to implement the ActionListener. If you implement the ActionListener class, you need to follow 3 steps:

1) Implement the ActionListener interface in the class:

ADVERTISEMENT

```java
public class ActionListenerExample Implements ActionListener
```

2) Register the component with the Listener:

```java
component.addActionListener(instanceOfListenerclass);
```

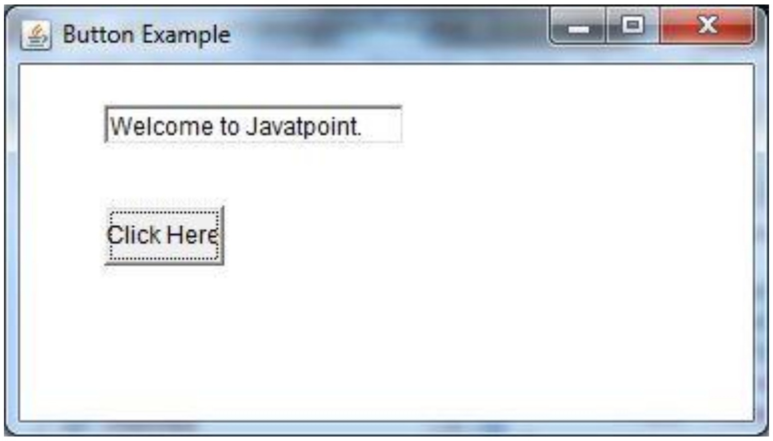
3) Override the actionPerformed() method:

```java
public void actionPerformed(ActionEvent e){
            //Write the code here
}
```

Java ActionListener Example: On Button click

```java
import java.awt.*;
import java.awt.event.*;
//1st step
public class ActionListenerExample implements ActionListener{
public static void main(String[] args) {
    Frame f=new Frame("ActionListener Example");
    final TextField tf=new TextField();
    tf.setBounds(50,50, 150,20);
    Button b=new Button("Click Here");
    b.setBounds(50,100,60,30);
    //2nd step
    b.addActionListener(this);
    f.add(b);
    f.add(tf);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
//3rd step
public void actionPerformed(ActionEvent e){
            tf.setText("Welcome to Javatpoint.");
}
}
```

Output:



What is the difference between execute(), executeQuery() and executeUpdate() methods in JDBC?

JDBCJava 8MySQLMySQLi Database

Once you have created the statement object you can execute it using one of the execute methods of the Statement interface namely, execute(), executeUpdate() and, executeQuery().

The execute() method: This method is used to execute SQL DDL statements, it returns a boolean value specifying weather the ResultSet object can be retrieved.

Example

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class Example {
   public static void main(String args[]) throws SQLException {
      //Registering the Driver
      DriverManager.registerDriver(new com.mysql.jdbc.Driver());
      //Getting the connection
      String mysqlUrl = "jdbc:mysql://localhost/sampleDB";
      Connection con = DriverManager.getConnection(mysqlUrl, "root", "password");
      System.out.println("Connection established......");
      //Creating the Statement
      Statement stmt = con.createStatement();
      //Executing the statement
      String createTable = "CREATE TABLE Employee( "
         + "Name VARCHAR(255), "
         + "Salary INT NOT NULL, "
         + "Location VARCHAR(255))";
      boolean bool = stmt.execute(createTable);

      System.out.println(bool);
   }
}
```

Output

```
Connection established......
false
```

executeUpdate(): This method is used to execute statements such as insert, update, delete. It returns an integer value representing the number of rows affected.

Example

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class ExecuteUpdateExample {
   public static void main(String args[]) throws SQLException {
      //Registering the Driver
      DriverManager.registerDriver(new com.mysql.jdbc.Driver());
      //Getting the connection
      String mysqlUrl = "jdbc:mysql://localhost/sampleDB";
      Connection con = DriverManager.getConnection(mysqlUrl, "root", "password");
      System.out.println("Connection established......");
      //Creating the Statement
      Statement stmt = con.createStatement();

      String insertData = "INSERT INTO Employee("
         + "Name, Salary, Location) VALUES "
         + "('Amit', 30000, 'Hyderabad'), "
         + "('Kalyan', 40000, 'Vishakhapatnam'), "
         + "('Renuka', 50000, 'Delhi'), "
         + "('Archana', 15000, 'Mumbai')";
      int i = stmt.executeUpdate(insertData);
      System.out.println("Rows inserted: "+i);
   }
}
```

Output

```
Connection established......
Rows inserted: 4
```

executeQuery(): This method is used to execute statements that returns tabular data (example select). It returns an object of the class ResultSet.

Example

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class ExecuteQueryExample {
   public static void main(String args[]) throws SQLException {
      //Registering the Driver
      DriverManager.registerDriver(new com.mysql.jdbc.Driver());
      //Getting the connection
      String mysqlUrl = "jdbc:mysql://localhost/sampleDB";
      Connection con = DriverManager.getConnection(mysqlUrl, "root", "password");
      System.out.println("Connection established......");
      //Creating the Statement
      Statement stmt = con.createStatement();

      //Retrieving data
      ResultSet rs = stmt.executeQuery("Select *from Employee");

      while(rs.next()) {
         System.out.print("Name: "+rs.getString("Name")+", ");
         System.out.print("Salary: "+rs.getInt("Salary")+", ");
         System.out.print("City: "+rs.getString("Location"));
         System.out.println();
      }
   }
}
```

Output

```
Connection established......
Name: Amit, Salary: 30000, City: Hyderabad
Name: Kalyan, Salary: 40000, City: Vishakhapatnam
Name: Renuka, Salary: 50000, City: Delhi
Name: Archana, Salary: 15000, City: Mumbai
```

A Password Field

Suppose we need a text field for entering a password.

As characters are typed in the field, we do not want them to be visible.

Solution: JPasswordField, a subclass of JTextField.

Constructors

JPasswordField() JPasswordField(String s)

JPasswordField(**int** w) JPasswordField(String s, **int** w)

Example

```
JPasswordField jpf =
        new JPasswordField("Enter your password here");
```

The echo character, which shows as the user types, is an asterisk (*) by default.

The echo character can be changed using:

```
jpf.setEchoChar('?');
```

If the enter (return) character is typed in the field, an ActionEvent is fired.

The contents of the field are provided by the method: jpf.getPassword(), which returns an array of **char**.

Sample Code

This program has a password field on a frame.

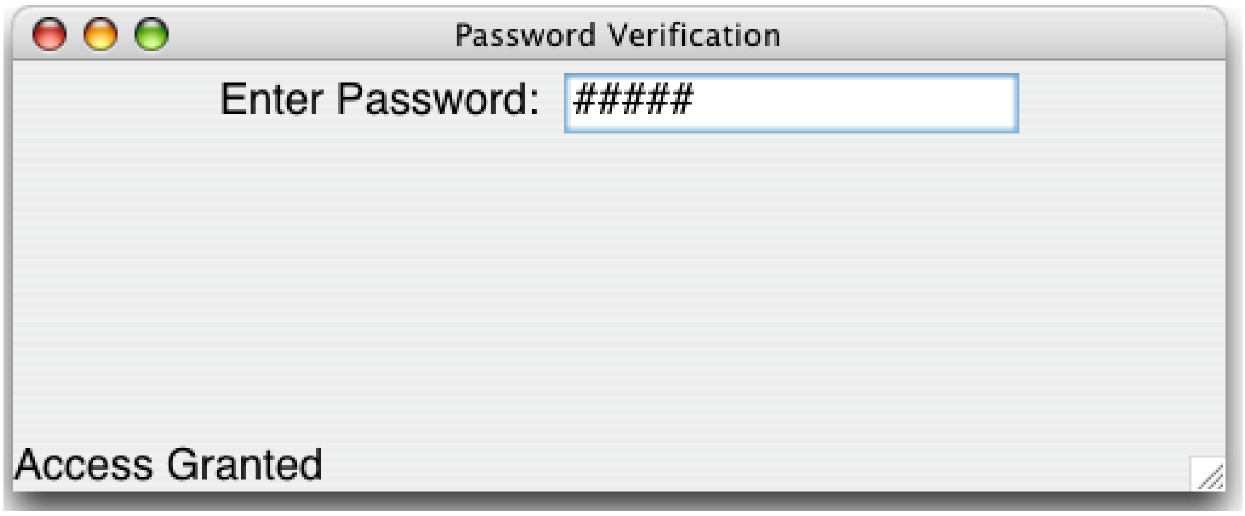
When return is typed in the field, it compares the string typed with a predefined password, “herky”.

```
import java.awt.*; import
java.awt.event.*; import
javax.swing.*;

public class JPApp extends JFrame

{

    private String pw = "herky"; private
    JPasswordField pwField; private JLabel
    ans;
```

```
JPApp()
{

    setTitle("Password  Verification");
    Font font = new Font("SanSerif", Font.PLAIN, 18); Container cp =
    getContentPane();

    JPanel panel = new JPanel();

    JLabel jl = new JLabel("Enter Password: "); jl.setFont(font);
    panel.add(jl);
    pwField = new JPasswordField(12);
    pwField.setFont(font); pwField.setEchoChar('#');
    panel.add(pwField);
    cp.add(panel, BorderLayout.CENTER);

    ans = new JLabel();
    ans.setFont(font);
    cp.add(ans,  BorderLayout.SOUTH);

    pwField.addActionListener(new   ActionListener()
    {

        public  void  actionPerformed(ActionEvent  e)

        {

            String password =
                                    new  String(pwField.getPassword());
            if  (pw.equals(password))
                ans.setText("Access   Granted");

            else
        }
                ans.setText("Access   Denied");
    } );

}
```

```
        void  getFocus()

        {

              pwField.requestFocus();
        }


        public static void main(String [] args)

        {

              JPApp jpa = new JPApp();

              jpa.addWindowListener(new   WindowAdapter()
              {

                    public  void  windowClosing(WindowEvent  e)

} );                {   System.exit(0);   }
```

```
                jpa.setSize(500,200);
                jpa.setVisible(true);
                jpa.getFocus();                    // give field the focus
        }
}
```



Password Verification

Enter Password: #####

Access Granted