Applet Programming

1. **What is an Applet? How applet program can differ from stand-alone program?**
   Or
   What is an Applet? How do applets differ from application programs?
   Or
   What are the differences between applet and application program?

Ans:

Applet:
Applets are small java programs that are primarily used in Internet computing. They can be transported over the Internet from one computer to another and run using the `Applet Viewer` or any Web browser that supports java. Like any application program, applet can perform arithmetic operations, display graphics, and accept user input and so on.

Difference between applet and application program:
Although both the applets and stand-alone applications are Java programs, there are significant differences between them.

- Applets do not use the `main()` method for initiating the execution of the code. Applets, when loaded, automatically call certain methods of `Applet` class to start and execute the applet code.
- Unlike stand-alone application, applets cannot be run independently. They are run from inside a Web page using a special feature known as HTML tag.
- Applets cannot read from or write to the files in the local computer.
- Applets cannot communicate with other servers on the network.
- Applets cannot run any program from the local computer.
- Applets are restricted from using libraries from other languages such as C or C++. (Remember, java language supports this feature through native methods).

2. What is local applet and remote applet?
Ans:

Local Applet:
An applet developed locally and stored in a local system is known as a *local applet*. When a Web page is trying to find a local applet, it does not need to use the Internet and therefore the local system does not require the Internet connection. It simply searches the directories in the local system and locates and loads the specific applet.

Remote Applet:
A *remote applet* is that which is developed by someone else and stored on a remote computer connected to the Internet. If our system is connected to the Internet, we can download the remote applet onto our system via the Internet and run it.

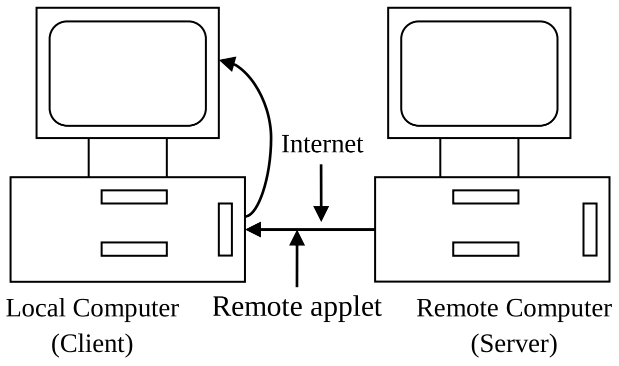

Fig: Loading a remote applet.

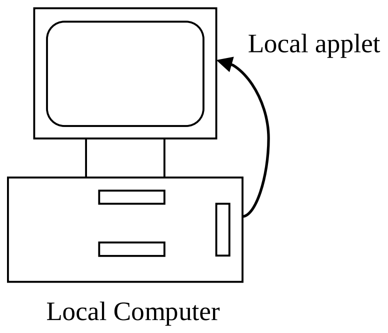

Fig: Loading a local applets

3. Describe the life cycle of an applet.
Or
Show the different stages in the lifecycle of an applet. (Transition diagram)
Ans:

The applet states include:
- Born on initialization state
- Running state
- Idle state
- Dead or destroyed state

Initialization State:
Applet enters the *initialization* state when it is first loaded. This is achieved by calling the `init()` method of Applet Class. The applet is born. At this stage, we may do the following:
- Create objects needed by the applet
- Set up initial values
- Load images or fonts
- Set up colors

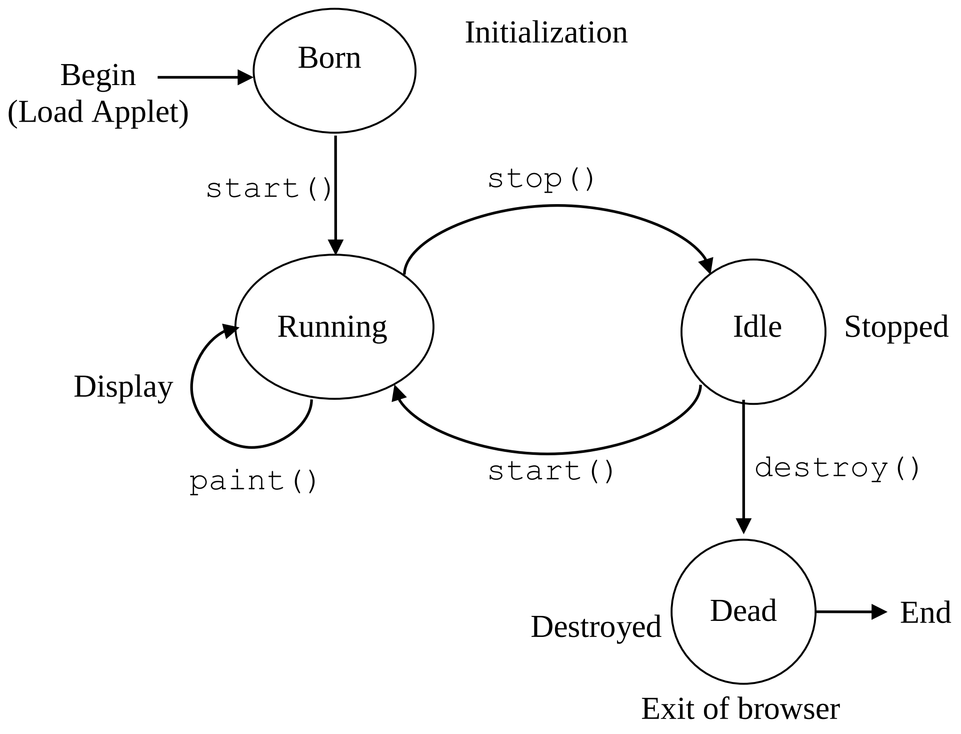

Fig: An applets state transition diagram.

Running State:
Applet enters the *running* state when the system calls the `start()` method of Applet Class. This occurs automatically after the applet is initialized. Starting can also occur if the applet is already in "stopped" (idle) state.

```
public void start(){
. . . . . . . . . .
. . . . . . . . . .  (Action)
. . . . . . . . . .
}
```

Idle or Stopped State:
An applet becomes *idle* when it is stopped from running. Stopping occurs automatically when we leave the page containing the currently running applet. We can also do so by calling the `stop()` method explicitly. If we use a thread to run the applet, then we must use `stop()` method to terminate the thread.

```
public void stop(){
. . . . . . . . . .
. . . . . . . . . .   (Action)
. . . . . . . . . .
}
```

Dead State:
An applet is said to be *dead* when it is removed from memory. This occurs automatically by invoking the `destroy()` method when we quit the browser.

```
public void destroy(){
. . . . . . . . . .
. . . . . . . . . .   (Action)
. . . . . . . . . .
}
```

Display State:
Applet moves to the *display* state whenever it has to perform some output operations on the screen. This happens immediately after the applet enters into the running state. The `paint()` method is called to accomplish this task.

```
public void paint(Graphics g){
. . . . . . . . . .
. . . . . . . . . .   (Display statements)
. . . . . . . . . .
}
```

4. Describe the purpose of the following Java methods in connection with applet.
i) start( ) ii) paint( ) iii) stop( ) iv) destroy( )

Ans:

Recommended to Question No. 3

5. How applet program can execute using HTML code?

Or

Write the HTML code to display an applet.

Ans:

1. Create a Java source file and save the program. (`.java` file)
2. Compile the source file (`.class` file)
3. Write a HTML document (`.html` file)
```
<html>
     <applet
          Code=HelloWorld.class
          Width=300
          Height=200>
     </applet>
</html>
```
4. Use the `appletviewer` to display the results.

6. Discuss the steps involved in developing and running a local applet.

Or

What are the steps involved in developing and testing an applet?

Ans:

The steps involved in developing and running a local applet are:

1. Building an applet code (`.java` file).
2. Creating an executable applet (`.class` file).
3. Designing a Web page using HTML tags.
4. Preparing `<APPLET>` tag.
5. Incorporating `<APPLET>` tag into Web page.
6. Creating HTML file.
7. Testing the applet code.

7. Write an applet program to draw a triangle, a rectangle, an ellipse and a circle.

Ans:

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

public class Draw extends Applet {
          public void paint(Graphics g){

               //drawing triangle
               Polygon t = new Polygon();
               t.addPoint(100, 100);
               t.addPoint(150, 150);
               t.addPoint(50,  150);
               g.drawPolygon(t);

               //drawing rectangle
               g.drawRect(10, 10, 30, 140);

               //drawing circle
               g.drawOval(150, 50, 90, 90);

               //drawing ellipse
               g.drawOval(100, 10, 75, 50);
          }
}
```

8. Write an applet program that will allow the user to enter two numeric values using textboxes and display their Sum. The program should allow the user to change the values without reloading the applet.

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

public class Sum extends Applet {
     TextField text1, text2;

     public void init() {
          text1 = new TextField(8);
          text2 = new TextField(8);
          add(text1);
          add(text2);
     }

     public void paint(Graphics g) {
          int x=0, y=0, z=0;
          String s1, s2, s3, s4;
          g.drawString("Input a number in each box", 20, 70);

          try {
               s1 = text1.getText();
               x = Integer.parseInt(s1);
               s2 = text2.getText();
               y = Integer.parseInt(s2);
          }

          catch (Exception ex) { }

          z=x+y;
          g.drawString("The Sum is= " + z,40,90);
     }

     public boolean action(Event event, Object object) {
          repaint();
          return true;
     }
}
```

9. Write an applet program that receives a numeric value as input from the user and then find its factorial and display the result on the screen. Also write an HTML page to test the applet and list the steps that should follow to run the applet program in a local computer.
Ans:

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

public class Factorial extends Applet {
     TextField text1;

     public void init() {
          text1=new TextField(8);
          add(text1);
     }

     public void paint(Graphics g) {
          int x=0;
          String s1;
          g.drawString("Input a number in box", 20, 70);

          try {
               s1=text1.getText();
               x=Integer.parseInt(s1);
          }

          catch (Exception ex) {}

          int sum=1;
          for (int i=1; i<=x; i++) {
               sum=sum*i;
          }

          g.drawString("Factorial of this number: "+sum,40,90);
     }

     public boolean action(Event event, Object object) {
          repaint();
          return true;
     }
}
```

HTML page:

```
<html>
     <applet
          Code=Factorial.class
          Width=300
          Height=200>
     </applet>
</html>
```

Run the applet Factorial using the following steps:

1. Create a Java source file and save the program. (`.java` file)
2. Compile the source file (`.class` file)
3. Write a HTML page (`.html` file) to test the applet.
4. Use the `appletviewer` to display the result.

10. Write an applet that allows the user to input values for the arguments required by method drawRect( ) and then draws a rectangle using the four input values. List the required steps to run the applet in a local computer.

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

public class RectDraw extends Applet {
     TextField text1, text2, text3, text4;

     public void init() {
          text1 = new TextField(8);
          text2 = new TextField(8);
          text3 = new TextField(8);
          text4 = new TextField(8);
          add(text1);
          add(text2);
          add(text3);
          add(text4);
     }

     public void paint(Graphics g) {
          int x = 0, y = 0, w = 0, h = 0;
          String s1, s2, s3, s4;
          g.drawString("Input a number in each box", 20, 70);

          try {
               s1 = text1.getText();
               x = Integer.parseInt(s1);
               s2 = text2.getText();
               y = Integer.parseInt(s2);
               s3 = text3.getText();
               w = Integer.parseInt(s3);
               s4 = text4.getText();
               h = Integer.parseInt(s4);
          }
          catch (Exception ex) {}

          g.setColor(Color.pink);
          g.fillRect(x, y, w, h);
     }

     public boolean action(Event event, Object object) {
          repaint();
          return true;
     }
}
```

HTML page:

```
<html>
     <applet
          Code=RectDraw.class
          Width=300
          Height=200>
     </applet>
</html>
```

Run the applet Factorial using the following steps:

1. Create a Java source file and save the program. (`.java` file)
2. Compile the source file (`.class` file)
3. Write a HTML page (`.html` file) to test the applet.
4. Use the `appletviewer` to display the result.

11. Write an applet program that receives a numeric value as input from the user and then decide is it odd/even and display the result on the screen. Also write an HTML page to test the applet and list the steps that should follow to run the applet program in a local computer.

Ans:

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

public class OddEven extends Applet {
     TextField text1;

     public void init() {
          text1=new TextField(8);
          add(text1);
     }

     public void paint(Graphics g) {
          int x=0;
          String s1;
          g.drawString("Input a number in box", 20, 70);

          try {
               s1=text1.getText();
               x=Integer.parseInt(s1);
          }

          catch (Exception ex) { }

          if (x%2==0)
               g.drawString("The Number is Even",40,90);
          else
               g.drawString("The Number is Odd",40,90);
     }

     public boolean action(Event event, Object object) {
          repaint();
          return true;
     }
}
```

12. Write an applet program that will allow the user to enter three numeric values using textboxes and display the largest one. The program should allow the user to change the values without reloading the applet.

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

public class Largest extends Applet {
      TextField text1, text2, text3;

      public void init() {
            text1 = new TextField(8);
            text2 = new TextField(8);
            text3 = new TextField(8);
            add(text1);
            add(text2);
            add(text3);
      }

      public void paint(Graphics g) {
            int x = 0, y = 0, z = 0;
            String s1, s2, s3, s4;
            g.drawString("Input a number in each box", 20, 70);

            try {
                  s1 = text1.getText();
                  x = Integer.parseInt(s1);
                  s2 = text2.getText();
                  y = Integer.parseInt(s2);
                  s3 = text3.getText();
                  z = Integer.parseInt(s3);
            }
            catch (Exception ex) {}
            if (x > y) {
              if (x > z)
                  g.drawString("Largest number is, x= " + x,40,90);
              else
                  g.drawString("Largest number is, z= " + z,40,90);
            }
            else {
              if (y > z)
                  g.drawString("Largest number is, y= " + y,40,90);
              else
                  g.drawString("Largest number is, z= " + z,40,90);
            }
      }
      public boolean action(Event event, Object object) {
            repaint();
            return true;
      }
}
```

13. Write an applet program that will allow the user to enter three numeric values using textboxes and display the smallest one. The program should allow the user to change the values without reloading the applet.

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

public class Smallest extends Applet {
      TextField text1, text2, text3;

      public void init() {
            text1 = new TextField(8);
            text2 = new TextField(8);
            text3 = new TextField(8);
            add(text1);
            add(text2);
            add(text3);
      }

      public void paint(Graphics g) {
            int x = 0, y = 0, z = 0;
            String s1, s2, s3, s4;
            g.drawString("Input a number in each box", 20, 70);

            try {
                  s1 = text1.getText();
                  x = Integer.parseInt(s1);
                  s2 = text2.getText();
                  y = Integer.parseInt(s2);
                  s3 = text3.getText();
                  z = Integer.parseInt(s3);
            }
            catch (Exception ex) {}
            if (x < y) {
              if (x < z)
                  g.drawString("Smallest number is, x= " + x,40,90);
              else
                  g.drawString("Smallest number is, z= " + z,40,90);
            }
            else {
              if (y < z)
                  g.drawString("Smallest number is, y= " + y,40,90);
              else
                  g.drawString("Smallest number is, z= " + z,40,90);
            }
      }
      public boolean action(Event event, Object object) {
            repaint();
            return true;
      }
}
```

14. What is `repaint()`? When do we call it?
Ans:

`repaint()`:
Whenever your applet needs to update the information displayed in its window, it simply calls `repaint()`.

The `repaint()` method is defined by an AWT. It causes the AWT run-time system to execute a call to your applets `update()` method, which, in its default implementation, calls `paint()`. Thus, for another part of your applet to output to its window, simply store the output and then call `repaint()`. The AWT will then execute a call to `paint()`, which can display the stored information. The method has four forms. The simplest version of repaint is:

```
void repaint();
```

This version causes the entire window to be repainted. The following version specifies the region that will be repainted:

```
void repaint(int left, int top, int width, int height);
```

15. What is the chain of actions after a call to repaint( ) method?
Ans:

http://mindprod.com/jgloss/repaint.html

16. How color can set to draw the background color of an object in an applet?
Ans:

To draw an object or text using a color object, you have to set the current color to be that color object, Use the `setColor()` method (a method for `Graphics` objects) to do this:

```
g.setColor(Color.green);
```

After you set the current color, all drawing operations will occur in that color.

In addition to setting the current color for the graphics context, you can also set the background and foreground colors for the applet itself by using the `Background()` and `setForeground()` methods.

The `setBackground()` method sets the background color of the applet, which is usually a light gray (to match the default background of the browser). It takes a single argument, a Color object:

```
setBackground(Color.white);
```

The `setForeground()` method also takes a single color as an argument, and it affects everything that has been drawn on the applet, regardless of the color in which it has been drawn. You can use `setForeground()` to change the color of everything in the applet at once, rather than having to redraw everything:

```
setForeground(Color.black);
```