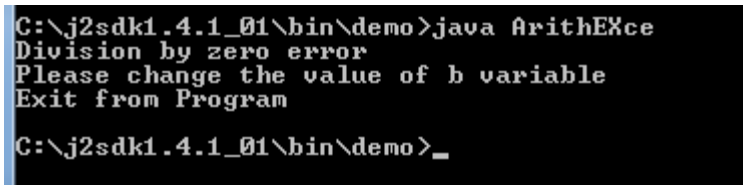
```
class ArithEXce
{

public static void main(String args[])
{

int a=10;
int b=0;

try
{
a = a/b;
System.out.println("Don't print");
}

catch(ArithmeticException e)
{
System.out.println("Division by zero error");
System.out.println("Please change the value of b variable");
}
System.out.println("Exit from Program");
}
}
```



```
import java.io.*;

class ExeTest
{

public static void main(String args[])
{

PrintWriter p = new PrintWriter(System.out,true);

for(int i=0;i<3;i++)
{
int j;

try
{
switch (i)
{

case 0:
int a=0;
j = 10/a;
break;

case 1:
char c[] = new char[3];
j = c[10];  //Array index out of Bound
break;

case 2:
char ch = "Java".charAt(7);
break;
```

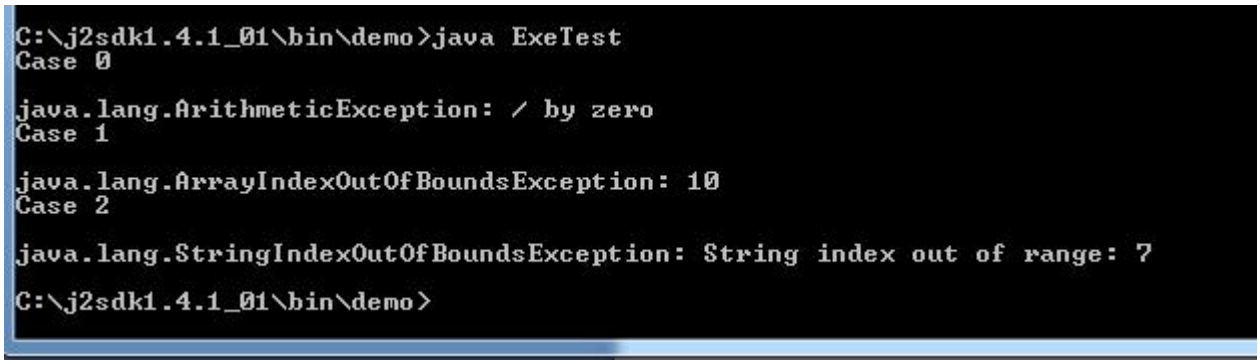
```
}

}

catch(Exception e)
{

System.out.println("Case " + i + "\n");
System.out.println(e);
}

}
}
}
```



init(), start(), stop(), and destroy()

The `init()` method is called exactly once in an applet's life, when the applet is first loaded. It's normally used to read PARAM tags, start downloading any other images or media files you need, and set up the user interface. Most applets have `init()` methods.

The `start()` method is called at least once in an applet's life, when the applet is started or restarted. In some cases it may be called more than once. Many applets you write will not have explicit `start()` methods and will merely inherit one from their superclass. A `start()` method is often used to start any threads the applet will need while it runs.

The `stop()` method is called at least once in an applet's life, when the browser leaves the page in which the applet is embedded. The applet's `start()` method will be called if at some later point the browser returns to the page containing the applet. In some cases the `stop()` method may be called multiple times in an applet's life. Many applets you write will not have explicit `stop()` methods and will merely inherit one from their superclass. Your applet should use the `stop()` method to pause any running threads. When your applet is stopped, it *should* not use any CPU cycles.

The `destroy()` method is called exactly once in an applet's life, just before the browser unloads the applet. This method is generally used to perform any final clean-up. For example, an applet that stores state on the server might send some data back to the server before it's terminated. many applets will not have explicit `destroy()` methods and just inherit one from their superclass.

For example, in a video applet, the `init()` method might draw the controls and start loading the video file. The `start()` method would wait until the file was loaded, and then start playing it. The `stop()` method would pause the video, but not rewind it. If the `start()` method were called again, the video would pick up where it left off; it

would not start over from the beginning. However, if `destroy()` were called and then `init()`, the video would start over from the beginning.

In the JDK's appletviewer, selecting the Restart menu item calls `stop()` and then `start()`. Selecting the Reload menu item calls `stop()`, `destroy()`, and `init()`, in that order. (Normally the byte codes will also be reloaded and the HTML file reread though Netscape has a problem with this.)

The applet `start()` and `stop()` methods are not related to the similarly named methods in the `java.lang.Thread` class.

Your own code may occasionally invoke `start()` and `stop()`. For example, it's customary to stop playing an animation when the user clicks the mouse in the applet and restart it when they click the mouse again.

Your own code can also invoke `init()` and `destroy()`, but this is normally a bad idea. Only the environment should call `init()` and `destroy()`.

```
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);

}
}
```

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

public class SampleBanner extends Applet implements Runnable {
  String str = "This is a simple Banner ";
  Thread t ;
  boolean b;

  public void init() {
    setBackground(Color.gray);
    setForeground(Color.yellow);
  }
  public void start() {
    t = new Thread(this);
    b = false;
    t.start();
  }
  public void run () {
    char ch;
    for( ; ; ) {
    try {
      repaint();
      Thread.sleep(250);
      ch = str.charAt(0);
      str = str.substring(1, str.length());
      str = str + ch;
    }
    catch(InterruptedException e) {}
    }
  }
  public void paint(Graphics g) {
    g.drawRect(1,1,300,150);
```

```
    g.setColor(Color.yellow);
    g.fillRect(1,1,300,150);
    g.setColor(Color.red);
    g.drawString(str, 1, 150);
  }
}
```

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

public class NewApplet extends Applet implements Runnable {
  String msg = " It is a moving Banner. ";
  char cha;
  boolean stopFlag = true;
  Thread t = null;

  public void start() {
    t = new Thread(this);
    stopFlag = false;
    t.start();
  }
  public void run() {
    for(;;) {
      try {
        repaint();
        Thread.sleep(250);
        cha = msg.charAt(0);
        msg = msg.substring(1,msg.length());
        msg = msg + cha;
        if(stopFlag) break;
      }
      catch(InterruptedException e) {}
    }
  }
  public void stop(){
    stopFlag = true;
    t = null;
  }
  public void paint(Graphics g) {
```

```
    g.drawString(msg,60,30);
  }
}
```