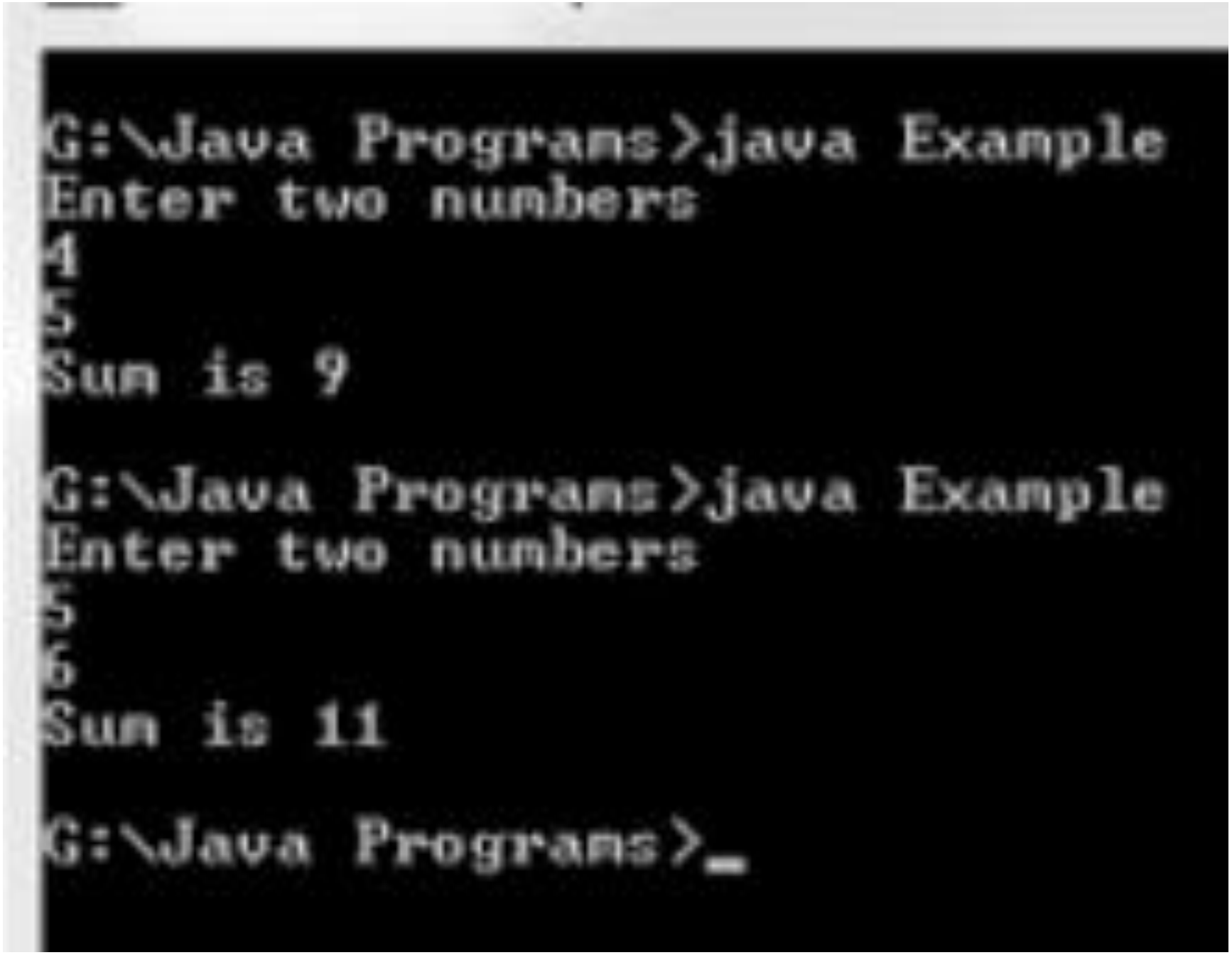
Introduction to GUI

- GUI : Graphical User Interface
- **Graphical user interface** is a type of user interface that allows users to interact with the screen using graphical components (Visual indicators) rather than text commands.

Console Based Application

```
import java.util.Scanner;
public class Example
{
 public static void main(String args[])
 {
  System.out.println("Enter two numbers");
  Scanner sc=new Scanner(System.in);
  int a=sc.nextInt();
  int b=sc.nextInt();
  int sum= a+b;
  System.out.println("Sum is "+sum);
 }
}
```


G:\Java Programs>java Example
Enter two numbers
4
5
Sum is 9
G:\Java Programs>java Example
Enter two numbers
5
6
Sum is 11
G:\Java Programs>_

Two APIs

- There are two sets of Java APIs for graphics programming:
  1. **AWT** (Abstract Windowing Toolkit)
  2. **Swing.**

AWT

- It consists of 12 packages
- only 2 packages - java.awt and java.awt.event - are commonly-used.

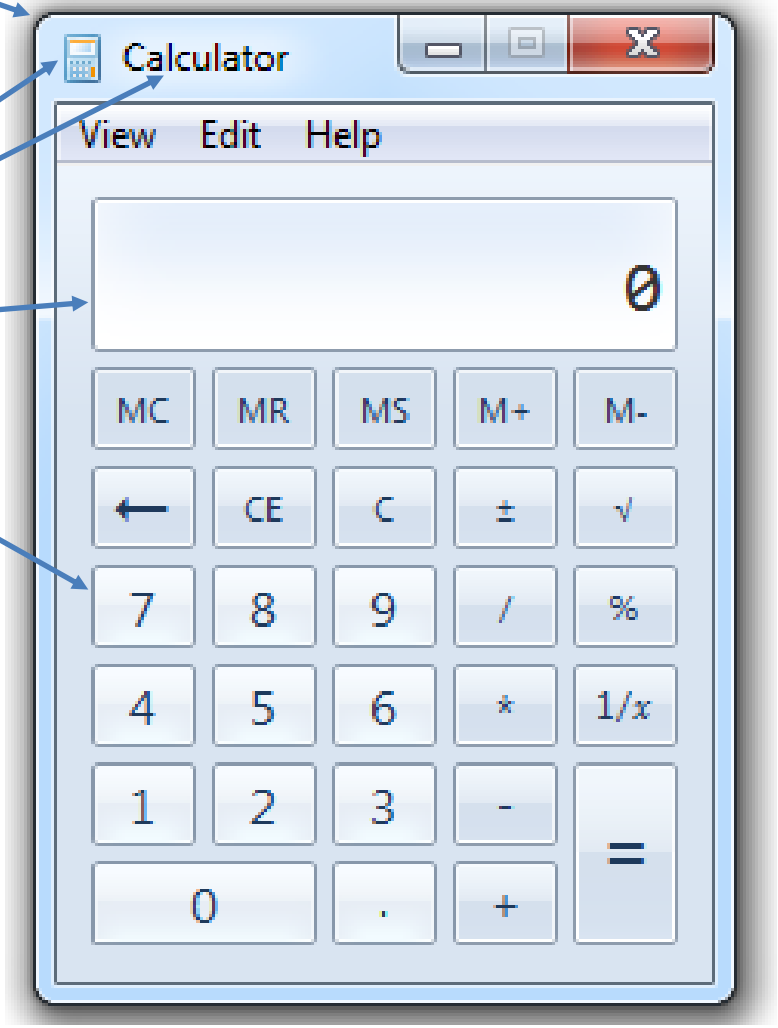
Classes in AWT

- The java.awt package contains the *core* AWT graphics classes
  - GUI Component classes (such as Button, TextField, and Label),
  - GUI Container classes (such as Frame, Panel, Dialog and ScrollPane),
  - Layout managers (such as FlowLayout, BorderLayout and Grid Layout),
  - Custom graphics classes (such as Graphics, Color and Font).

Rectangular Frame acts as basic container for components

Components placed within frame – buttons, lables, text fields, icons etc

Layout (optional) defines the style of arranging Components on frames



AWT events

- The java.awt.event package supports event handling
  - Event classes (such as ActionEvent, MouseEvent, KeyEvent and WindowEvent),
  - Event Listener Interfaces (such as ActionListener, MouseListener, KeyListener and WindowListener)

Container

- A Frame is the *top-level container* of an AWT GUI program
  - A Frame has a title bar (containing an icon, a title, and the minimize/maximize(restore-down)/close buttons), an optional menu bar and the content display area.
- A Panel is a *rectangular area* (or partition) used to group related GUI components.

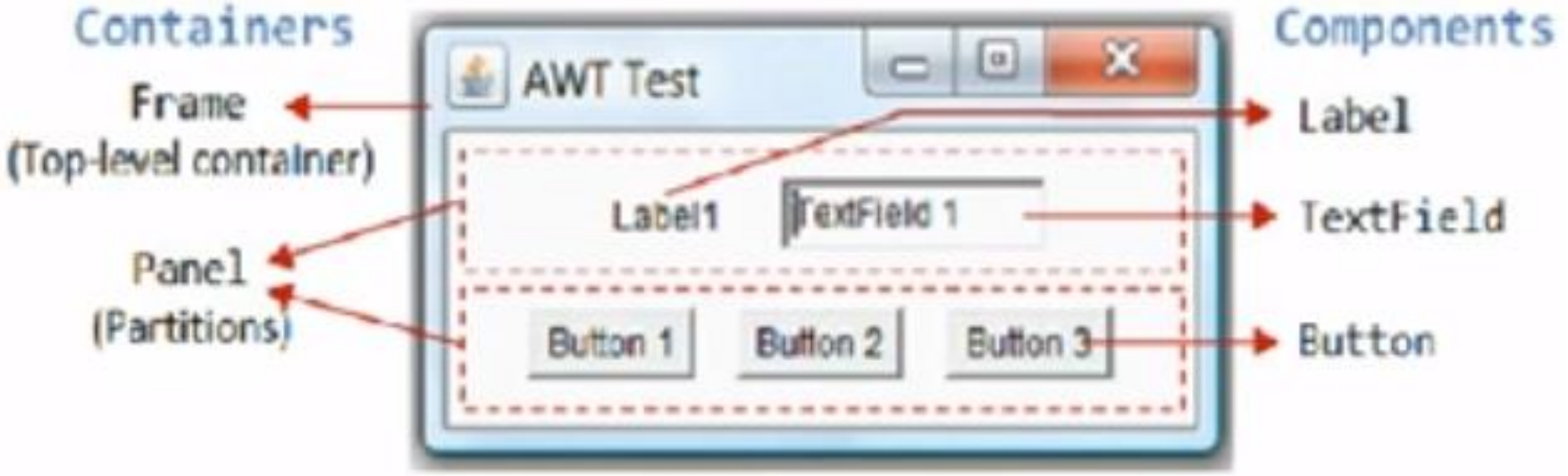
...Container

- In a GUI program, a component must be kept in a container.
- Every container has a method called add(Component c)

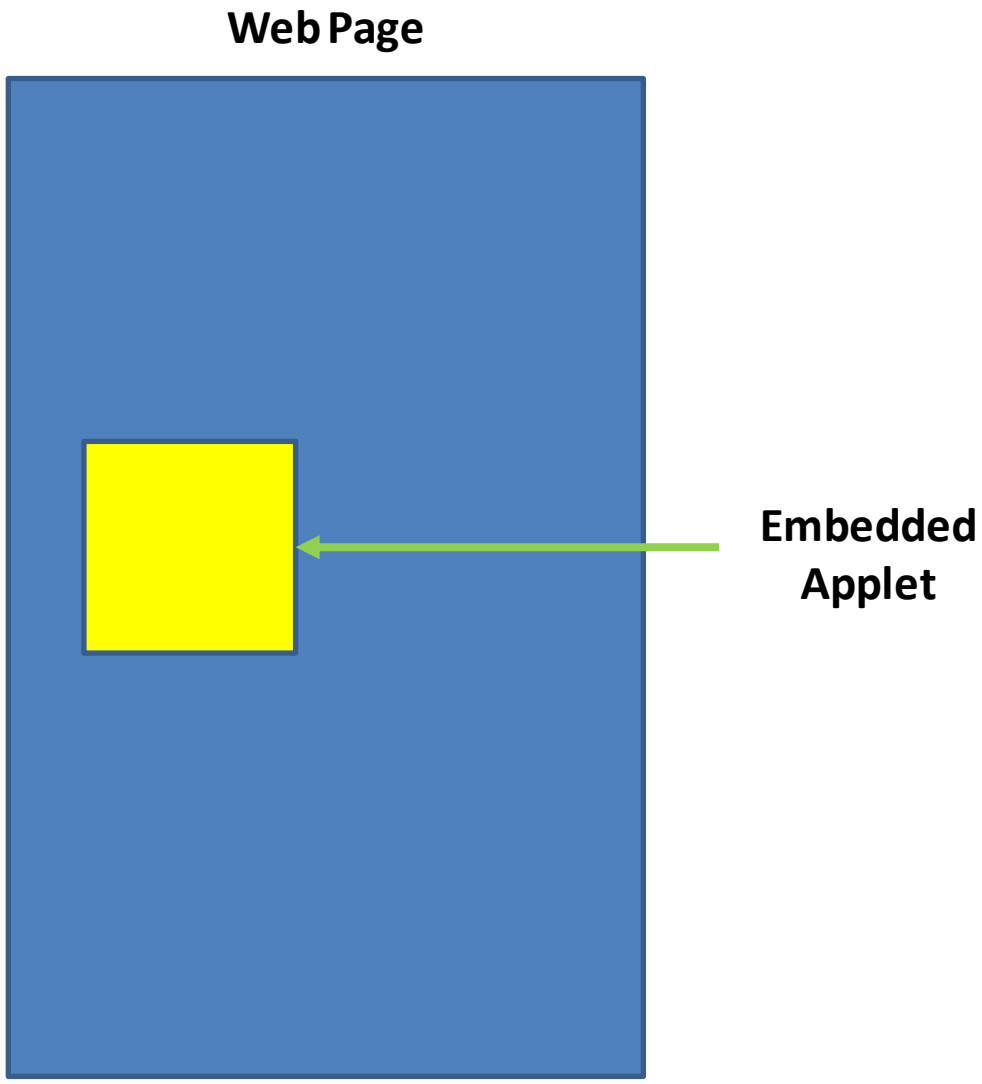
Container Classes

- Each GUI program has a *top-level container*. The commonly-used top-level containers in AWT are Frame, Dialog and Applet:


Enter your name here
TextField
Click Me!
Button
This is Label
Label
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
List
Red
Red
Green
Blue
Choice
one
two
three
CheckBox
Alpha
Beta
Charlie
CheckBoxGroup


Containers
Frame
(Top-level container)
Panel
(Partitions)
AWT Test
Label1
TextField 1
Button 1
Button 2
Button 3
Components
Label
TextField
Button

Applets in Java


Web Page
Embedded
Applet

What is an applet?

- An applet is a Java program that runs in a Web browser.
- Applet is a container class like Frame
- An applet is a Java class that extends the java.applet.Applet class
- Applets are designed to be embedded within an HTML page

What is an applet?

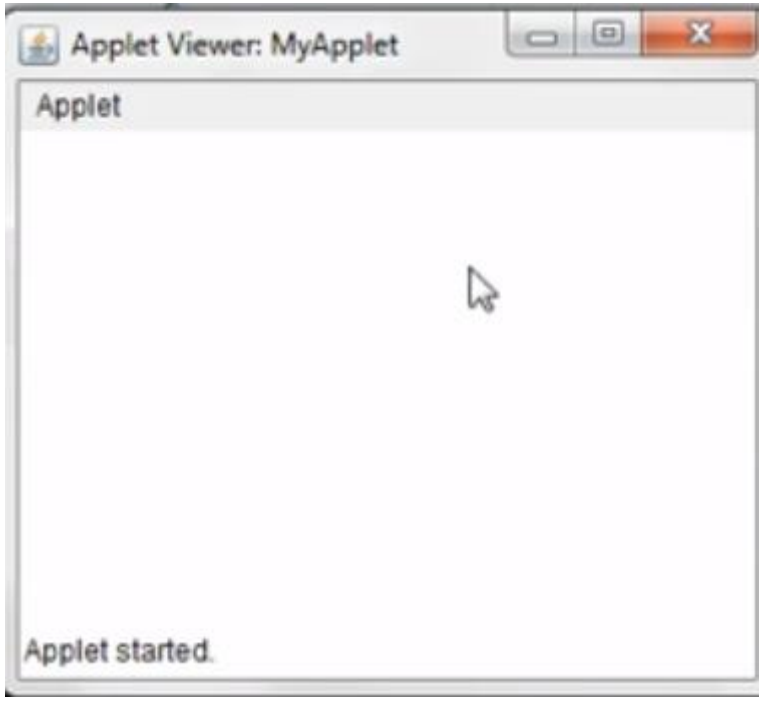
- When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine
- A JVM is required to view an applet
- The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime

Practical

- How to make an applet program?
- How a programmer can compile and run an applet code?
- How applet can be embedded in the HTML code?

```java
import java.applet.Applet;
/* <applet code="MyApplet" width="300" height="200"> </applet> */
public class MyApplet extends Applet{

}
```



Life Cycle of an Applets

What is an applet?

- An applet is a Java program that runs in a Web browser.
- An applet is a Java class that extends the java.applet.Applet class
- Applets are designed to be embedded within an HTML page

Applet's Life Cycle

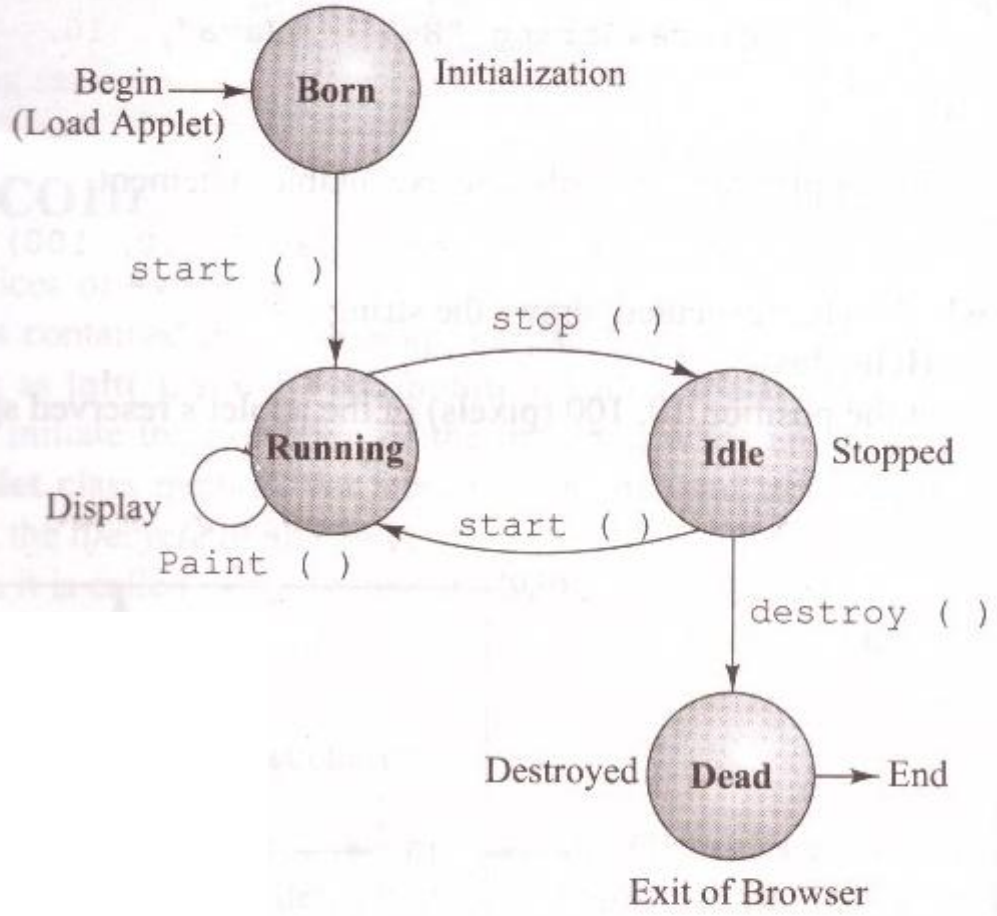
- JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's life time

- Four methods in the Applet class give you the framework on which you can build any Applet application

APPLET LIFE CYCLE

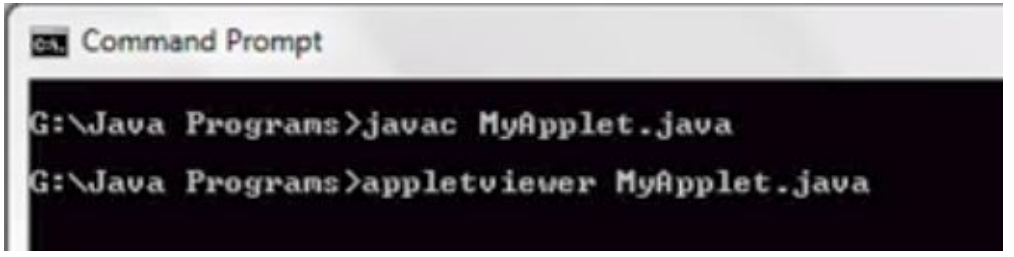
Every Java applet inherits a set of default behaviours from the **Applet** class. As a result, when an applet is loaded, it undergoes a series of changes in its state as shown in Fig. 14.5. The applet states include:

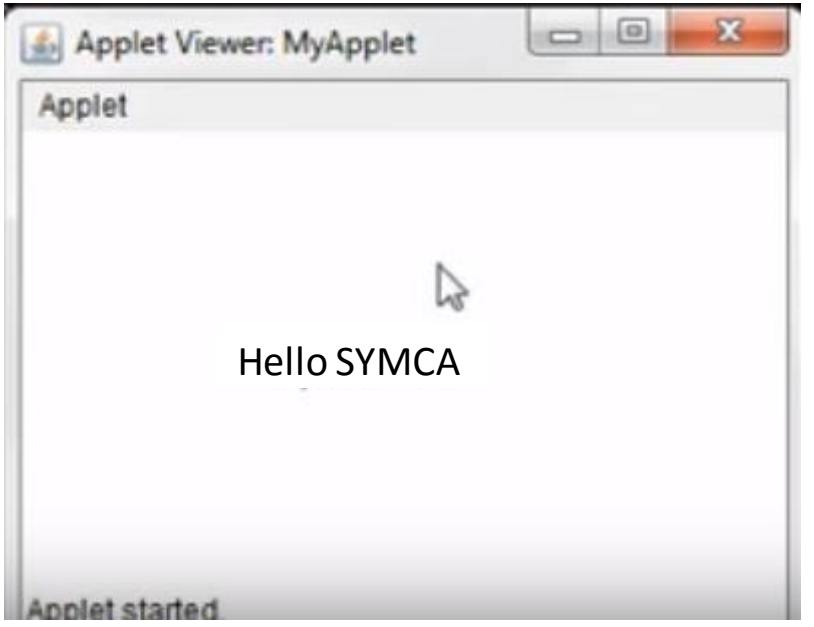
- Born on initialization state
- Running state
- Idle state
- Dead or destroyed state



An applet's state transition diagram

```java
import java.applet.Applet;
import java.awt.*;
/* <applet code="MyApplet" width="300" height="200"> </applet> */
public class MyApplet extends Applet{
    public void paint(Graphics g){
      g.drawString(" Hello SYMCA ",100,100);
    }
}
```





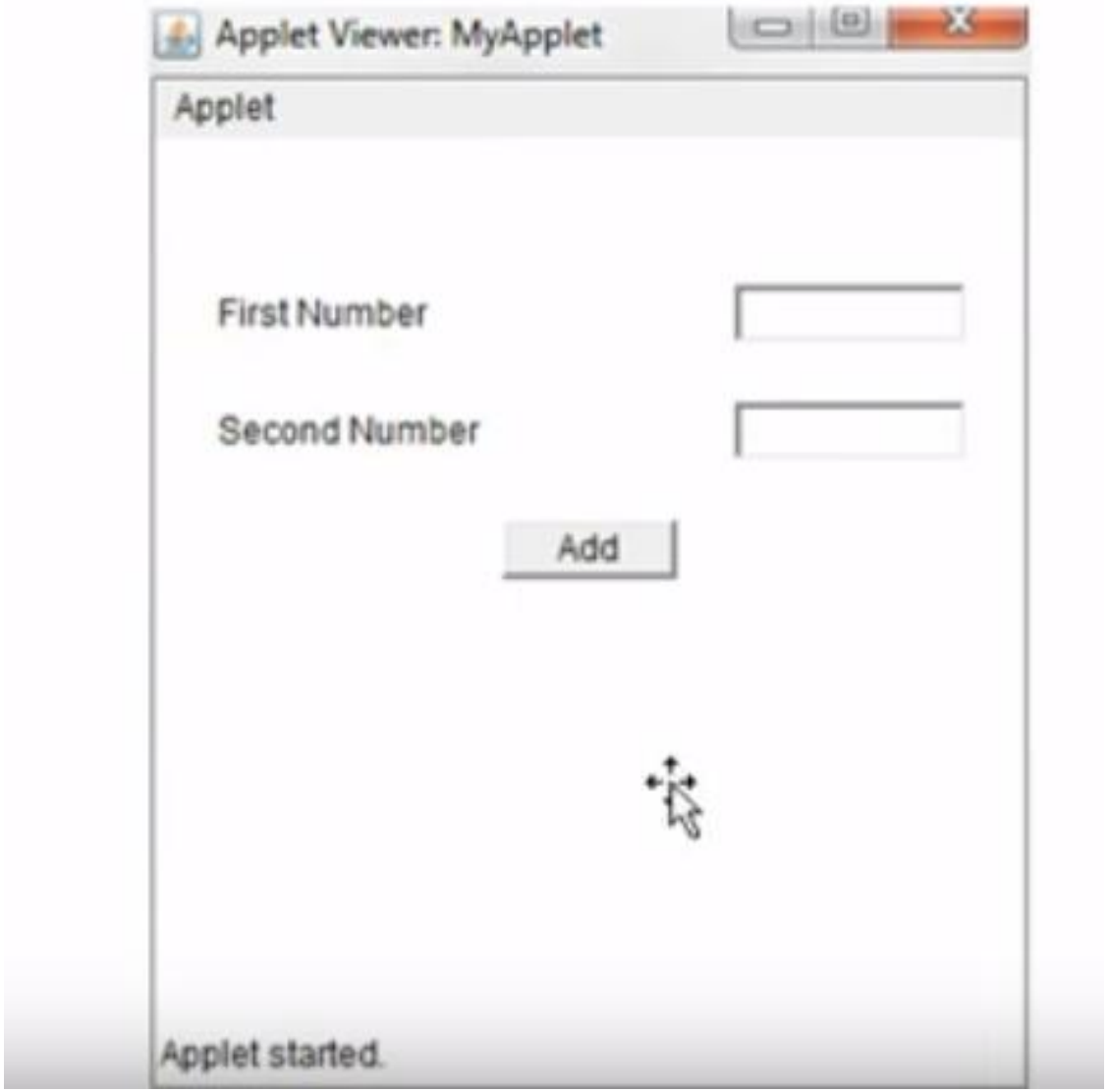
Components

- ❑ Button
- ❑ TextField
- ❑ Label
- ❑ Checkbox
- ❑ Choice
- ❑ List

Component on applet

- We need to design GUI so that user can interact with the applet
- We have to place components on the applet container, to meet this requirement

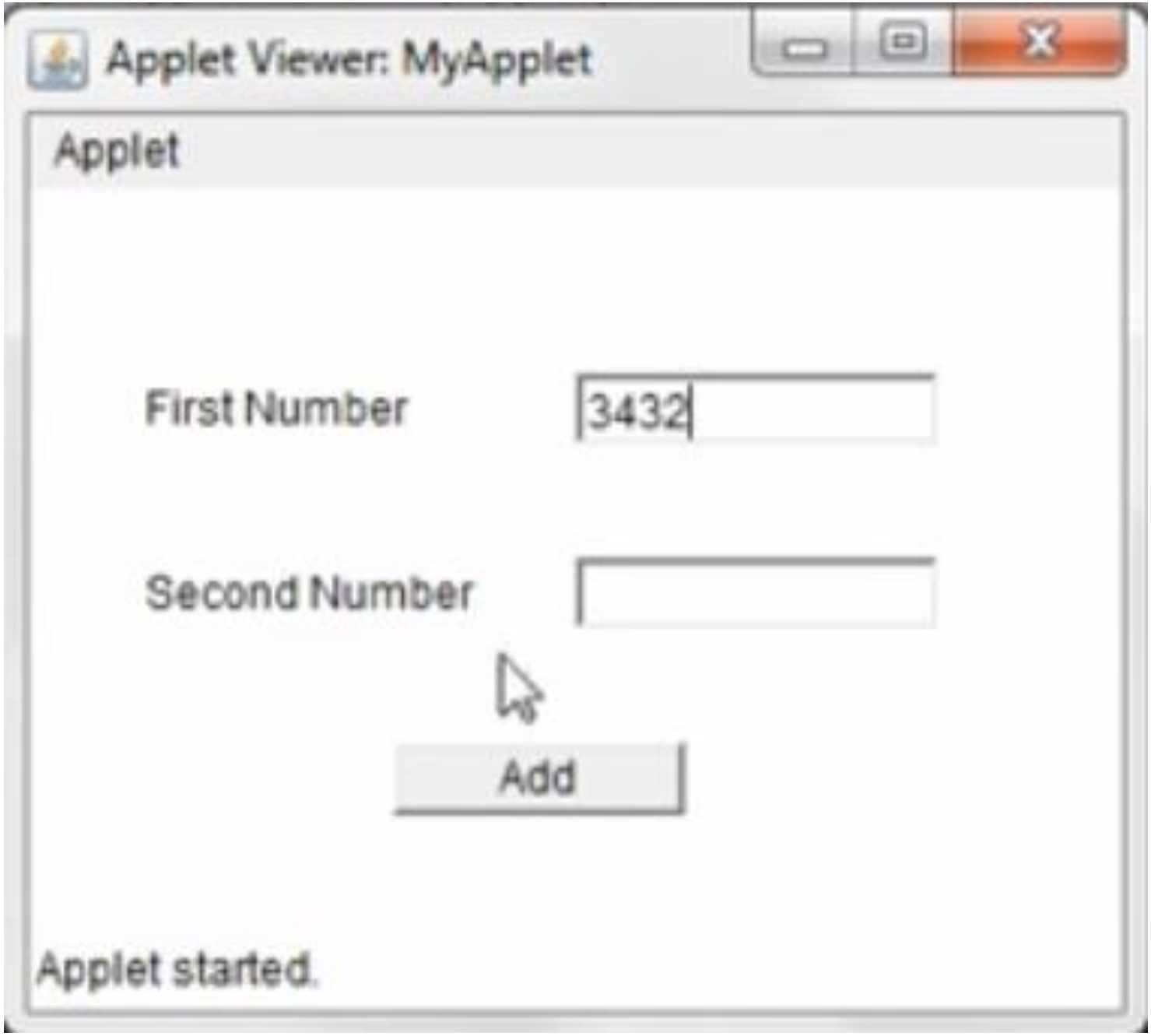
Example target



Need to learn

- Basic skeleton of an applet
- Creating Component reference variables
- Initialize Components
- Adding components to the applet
- Setting layout

```
import java.applet.Applet;
import java.awt.*;
/* <applet code="MyApplet" width="300" height="200"> </applet> */
public class MyApplet extends Applet{
    Label l1,l2;
    TextField t1,t2;
    Button b1;
    public void init(){
    l1=new Label("First Number");
    l2=new Label("Second Number");
    t1=new TextField();
    t2=new TextField();
    b1=new Button("Add");
    setLayout(null);
    l1.setBounds(30,50,100,20);
    l2.setBounds(30,100,100,20);
    t1.setBounds(150,50,100,20);
    t2.setBounds(150,100,100,20);
    add(t1);
    add(l1);
    add(l2);
    add(t2);
    add(b1);
    }
```


Applet Viewer: MyApplet
Applet
First Number
3432
Second Number
Add
Applet started.

Event Handling in an Applet

Event

- Changing the state of an object is known as an event.
- For example, click on button, dragging mouse, minimizing window, getting focus on component, mouse over on component, etc.

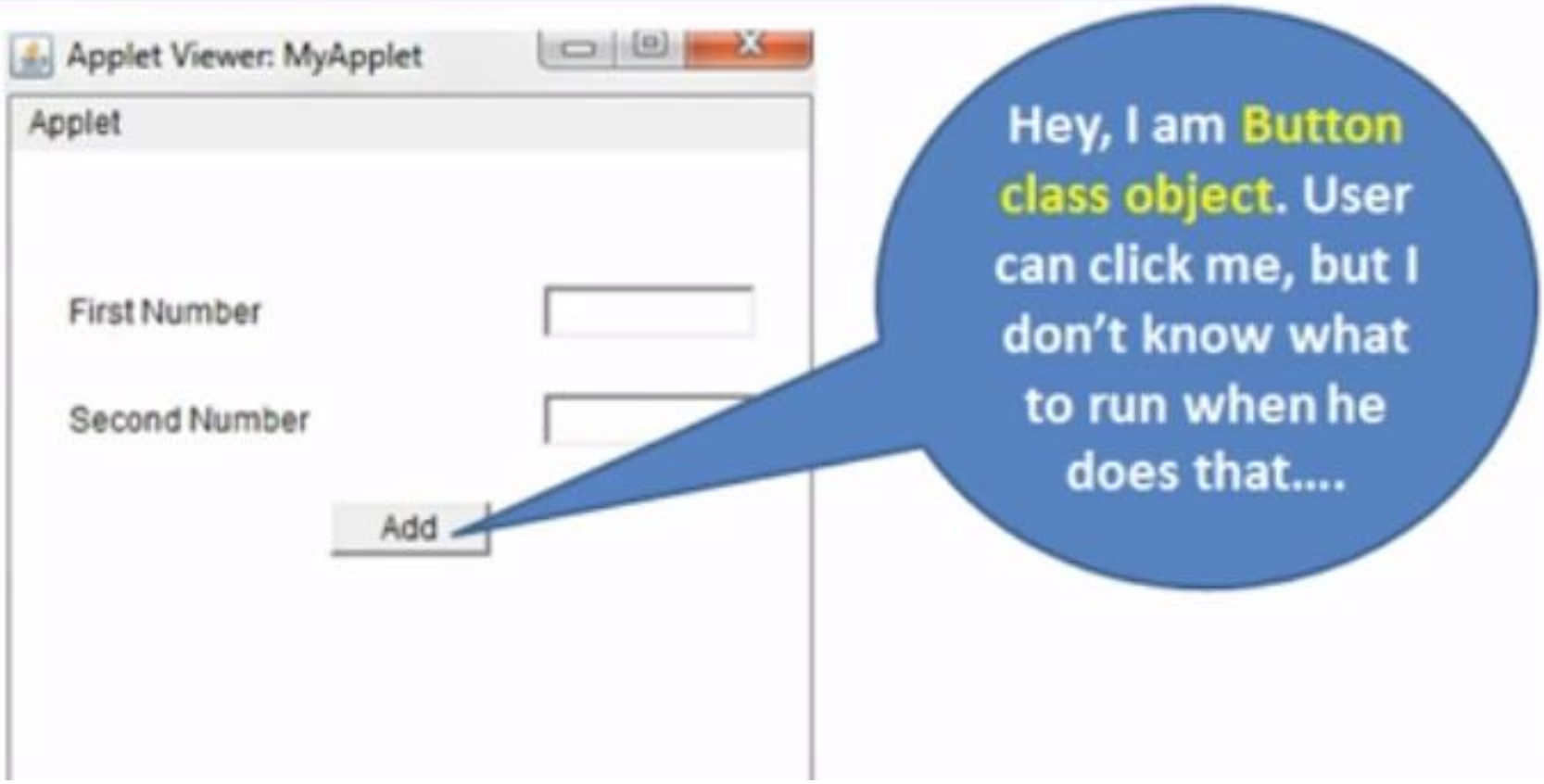
java.awt.event

- The java.awt.event package provides many event classes and Listener interfaces for event handling

Event Handling

- Event handling is to make java code ready to respond any particular event.

Feelings of a button



Feelings of the programmer(You)

I am the programmer, I have to write code that should be executed on click of a button

```
public void someFunction()
{

  //Write handling code here

}
```

Two ends of the story

Hey Programmer, I have a method to register your code with me

Ok that means button.addActionListener();

Communication can solve the issue

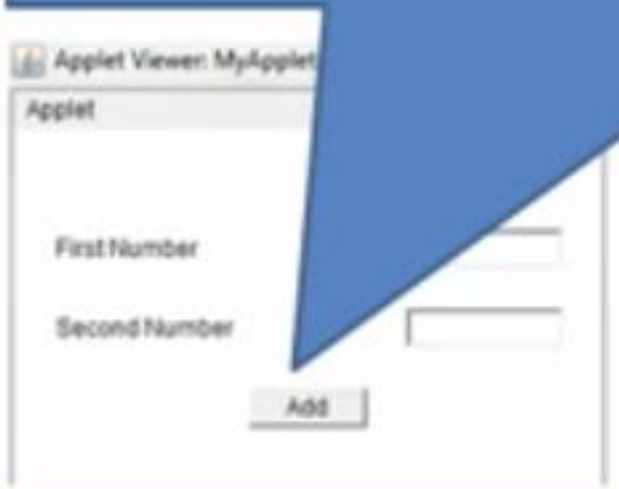
Yes, but you have to pass an object to this function

button.addActionListener(here?);
But object of which class?

Conversation continues…

See prototype defined in Button class is as

public void addActionListener(ActionListener actionListener)



But ActionListener is an interface, which means I have to pass an object of class which implements ActionListener.

button.addActionListener(object);

Ok, then I have to make a class which implements ActionListener

```
Class MyHandler implements ActionListener
{
public void someFunction(ActionEvent e)
{

  //Write handling code here
}
}
```

Ok, then I have to make a class which implements ActionListener

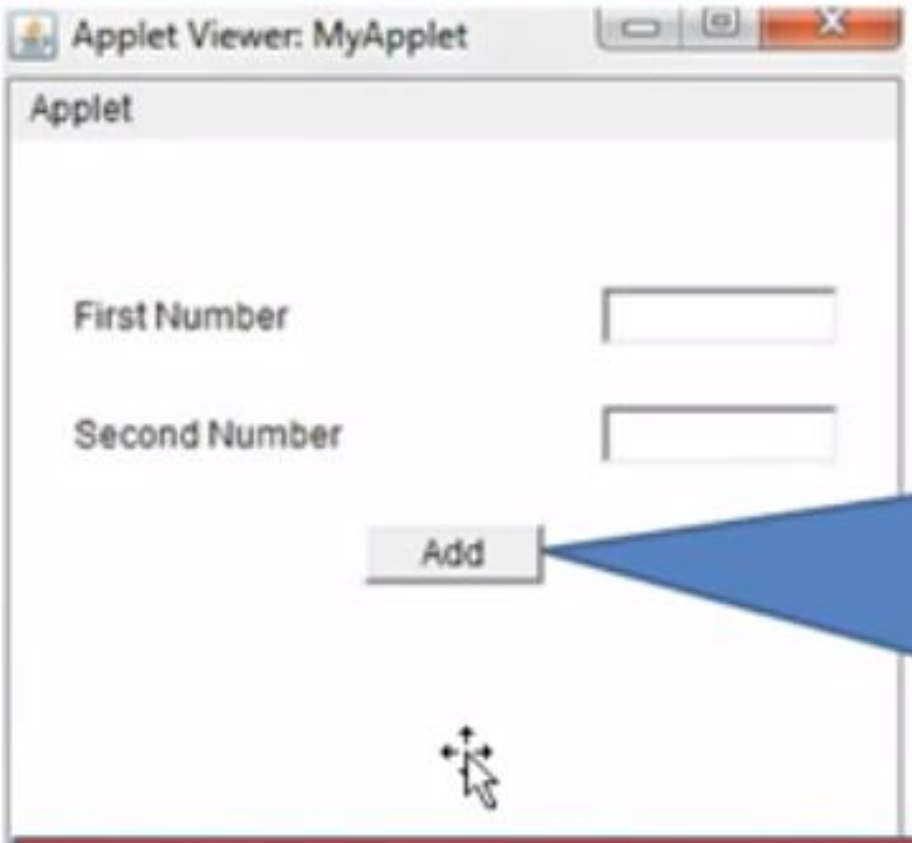
```
class MyHandler implements ActionListener
{
public void someFunction(ActionEvent e)
{

  //Write handling code here
}
}
```

Got it

```
{
   ...
  button.addActionListener(new MyHandler());
}
```

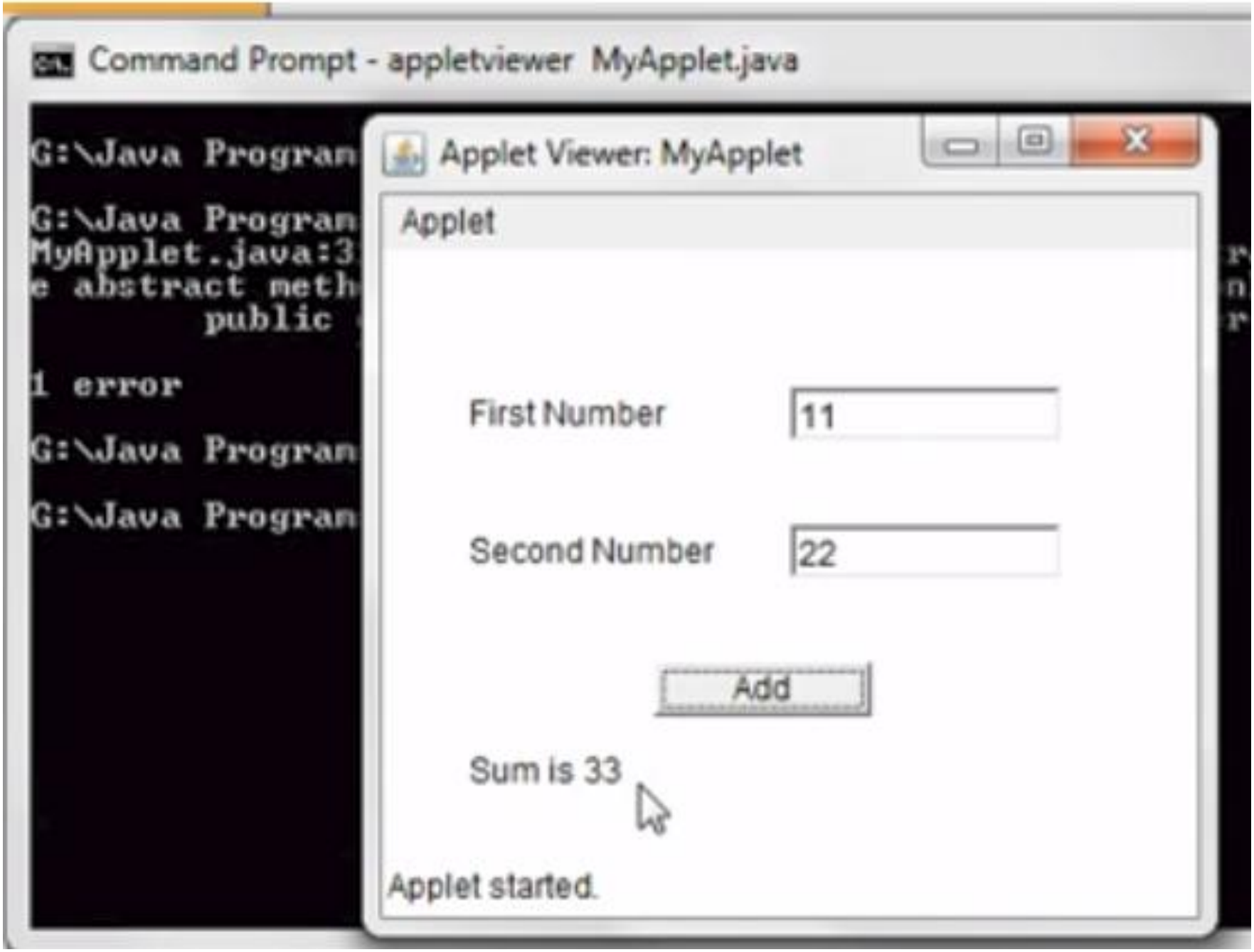
concept



Now I have an object of MyHandler referenced by actionListener(a Reference variable of ActionListener)

We know that reference variable can invoke only those members of referred object whose prototype belongs to the same interface.

```
add(t1);
add(l1);
add(l2);
add(t2);
add(button);
add(l3);
button.addActionListener(new MyHandler());
}
public class MyHandler implements ActionListener
{
  public void actionPerformed(ActionEvent e){
   int a,b,s;
   a=Integer.parseInt(t1.getText());
   b=Integer.parseInt(t2.getText());
   s=a+b;
   l3.setText("Sum is "+s);
  }
}
```

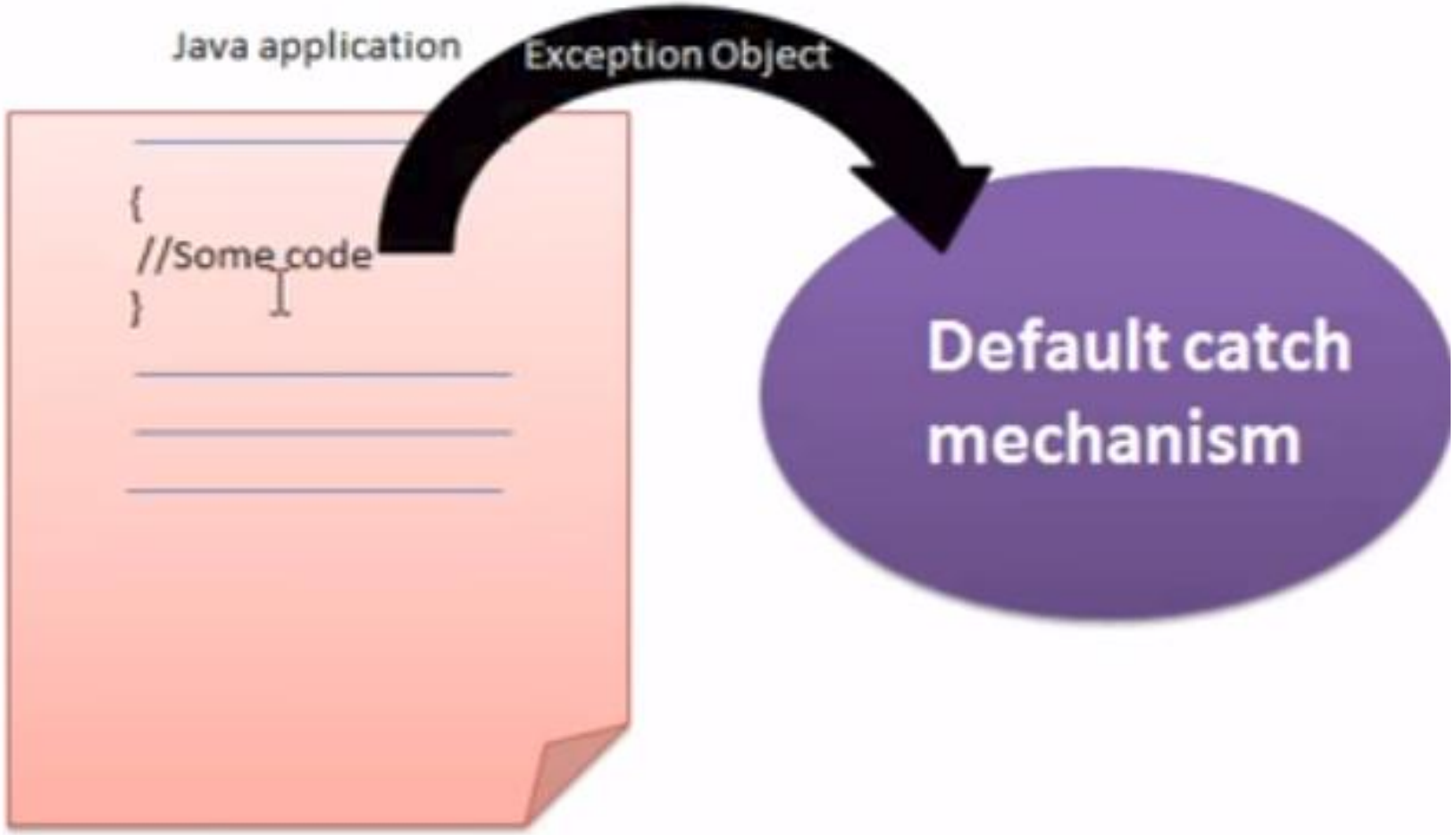

Command Prompt - appletviewer MyApplet.java
G:\Java Progran
G:\Java Progran
MyApplet.java:3
e abstract meth
public
1 error
G:\Java Progran
G:\Java Progran
Applet Viewer: MyApplet
Applet
First Number
11
Second Number
22
Add
Sum is 33
Applet started.

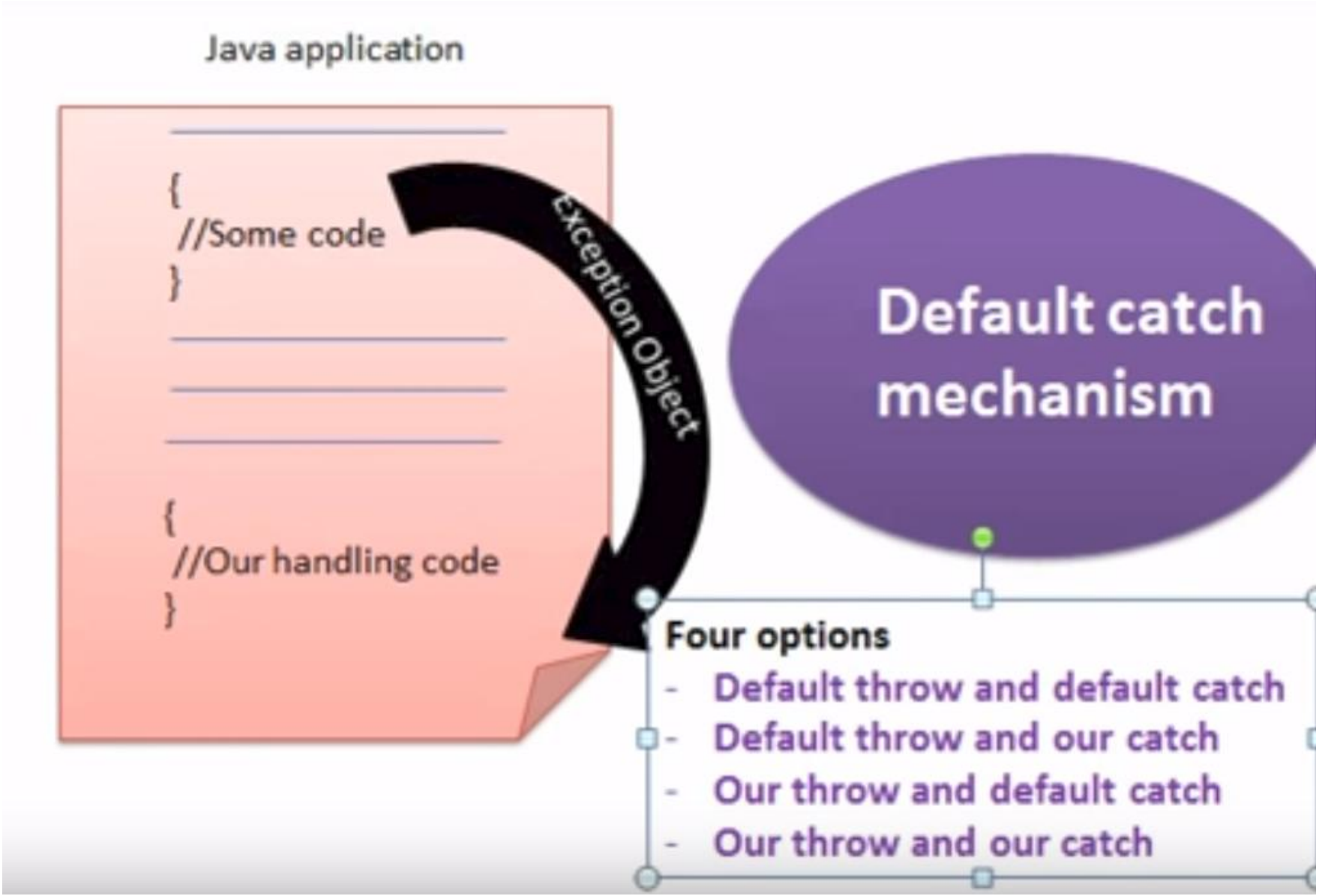
Exception handling in Java

What is an Exception?

- **Exceptions in java** are any abnormal, unexpected events or extraordinary conditions that may occur at runtime

So what is an exception?

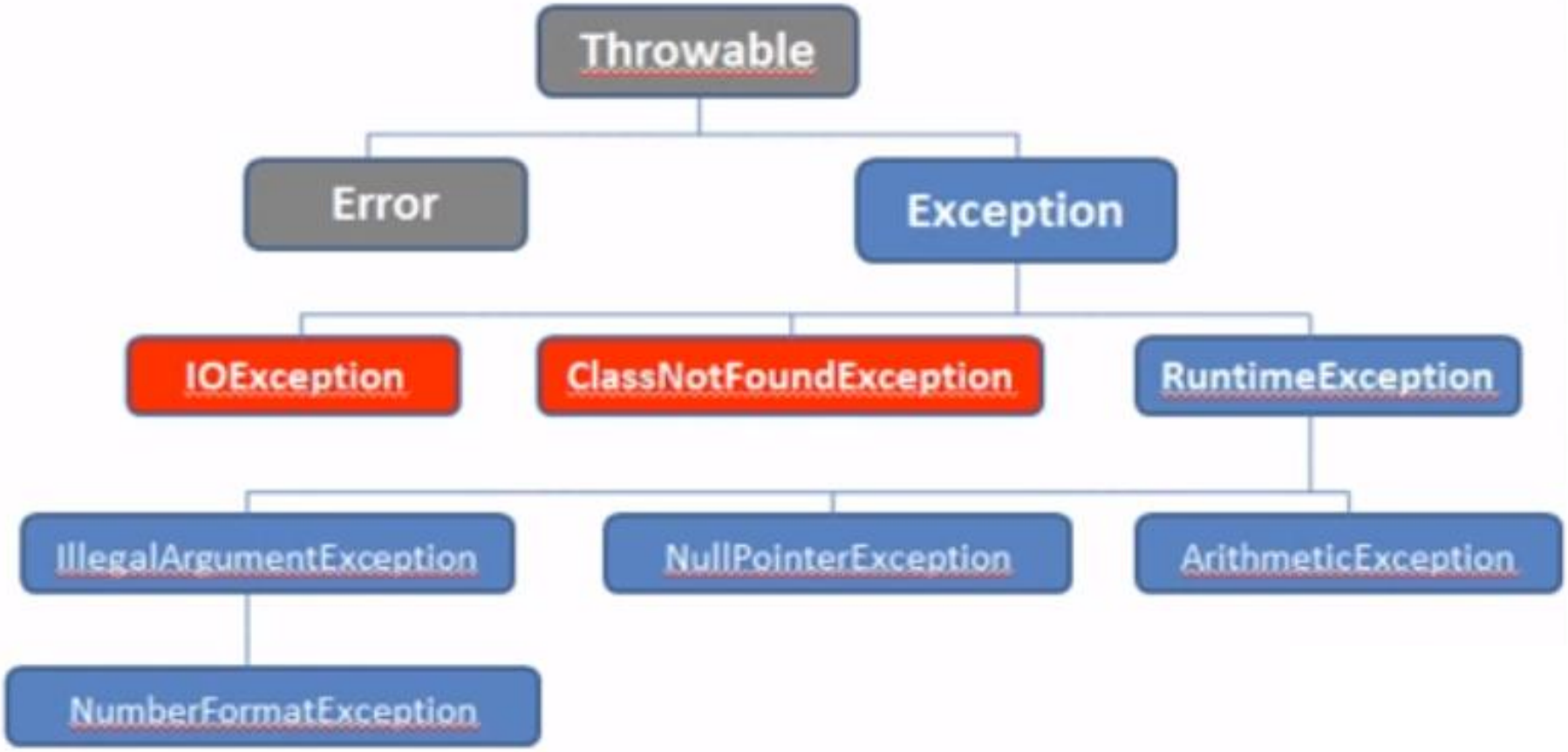



Java application
{
//Some code
}
{
//Our handling code
}
Exception Object
Default catch mechanism
Four options
- Default throw and default catch
- Default throw and our catch
- Our throw and default catch
- Our throw and our catch

Exception Handling

- Java exception handling is used to handle error conditions in a program systematically by taking the necessary action

Class Hierarchy



throw and catch

- Java exceptions are raised with the throw keyword and handled within a catch block.

java

```java
class Example{
 public static void main(String[] args){
  System.out.println("Result: "+3/0);
 }
}
```

Command Prompt

```
G:\Java Programs>javac Example.java

G:\Java Programs>java Example
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Example.main(Example.java:3)

G:\Java Programs>
```

```java
class Example{
 public static void main(String[] args){
    String s1=null;
     System.out.println("First Line");
   System.out.println("String length is "+s1.length());
    System.out.println("Last Line");
 }
}
```

```
G:\Java Programs>java Example
First Line
Exception in thread "main" java.lang.NullPointerException
        at Example.main(Example.java:5)

G:\Java Programs>
```

Throwable

- The Throwable class provides a String variable that can be set by the subclasses to provide a detail message that provides more information of the exception occurred
- All classes of Throwables define a one-parameter constructor that takes a string as the detail message
- The class Throwable provides getMessage() function to retrieve an exception

Unchecked Exception handling in Java

Exceptions are of two types

- The class Exception represents exceptions that a program faces due to abnormal or special conditions during execution.
- Exceptions can be of 2 types: **Checked** (Compile time Exceptions)/ **Unchecked** (Run time Exceptions).

Unchecked Exceptions

- **Unchecked exceptions** are RuntimeException and any of its subclasses
- ArrayIndexOutOfBounds, NullPointerException and so on are all subclasses of the java.lang.RuntimeException class, which is a subclass of the Exception class.

Four ways

- Default throw and default catch
- Default throw and our catch
- Our throw and default catch
- Our throw and our catch

Default throw and our catch

```
try
{
    <code>
} catch (<exception type> <parameter>) {
    // 0 or more <statements>
}
 finally {
// finally block <statements>
}
```

```java
class Example{
 public static void main(String[] args){
    try{
     System.out.println(3/0);
     System.out.println("In try");
    }
    catch(ArithmeticException e){
     System.out.println("Exception: "+e.getMessage());
    }
    System.out.println("Hello");
 }
}
```

```
G:\Java Programs>javac Example.java

G:\Java Programs>java Example
Exception: / by zero
Hello

G:\Java Programs>_
```

```
class Example{
 public static void main(String[] args){
    try{
     System.out.println(3/0);
     System.out.println("In try");
    }
    catch(NullPointerException e)
    {
     System.out.println("Exception: "+e.getMessage());
    }
    catch(ArithmeticException e){
     System.out.println("Exception: "+e.getMessage());
    }
    System.out.println("Hello");
 }
}
```

Remember

- For each try block there can be zero or more catch blocks, but only one finally block
- The catch blocks and finally block must always appear in conjunction with a try block
- A try block must be followed by either at least one catch block or one finally block.
- The order exception handlers in the catch block must be from the most specific exception

Explicit throw

- A program can explicitly throw an exception using the throw statement besides the implicit exception thrown.

- Syntax:
  - throw <throwableInstance>;

- throw <throwableInstance>;

- The Exception reference must be of type Throwable class or one of its subclasses
- A detail message can be passed to the constructor when the exception object is created.

```java
class Example{
 public static void main(String[] args){
    int balance=5000;
    int withdrawlAmount=6000;

    if(balanace < withdrawlAmount)
      throw new ArithmeticException("Insufficient balance");

    balance=balance-withdrawlAmount;
    System.out.println("Transaction Successfully completed");
    System.out.println("Program continue...");
 }
}
```

```
G:\Java Programs>javac Example.java

G:\Java Programs>java Example
Exception in thread "main" java.lang.ArithmeticException: Insufficient bala
        at Example.main(Example.java:7)

G:\Java Programs>
```

```java
class Example{
 public static void main(String[] args){
   int balance=5000;
   int withdrawlAmount=6000;
   try
   {
    if(balance < withdrawlAmount)
      throw new ArithmeticException("Insufficient balance");
    balance=balance-withdrawlAmount;
    System.out.println("Transaction Successfully completed");
   }
   catch(ArithmeticException e)
   {
    System.out.println("Exception: "+e.getMessage());
   }
   System.out.println("Program continue...");
 }
}
```

```
G:\Java Programs>javac Example.java

G:\Java Programs>java Example
Exception: Insufficient balance
Program continue...

G:\Java Programs>
```

One question

- Why should we throw an exception object?
  - Because we want to set a different message
  - Because java cannot recognize exceptional situation of business logic

Compile Time Error in checked exception

- Checked Exceptions forces programmers to deal with the exception that may be thrown
- IOException, SQLException, IllegalThreadStateException, etc are checked exceptions
- "checked" means they will be checked at compile time itself

throws

- A **throws** clause can be used in the method prototype

```
Method() throws <ExceptionType1>,…,
    <ExceptionTypen>
 {


 }
```

```
import java.io.*;
public class Example
{
 public static void main(String []args)
 {
  throw new IOException();
  System.out.println("After Exception");
 }
}
```


Command Prompt
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Saurabh>g:
G:\>cd "Java Programs"
G:\Java Programs>javac Example.java
Example.java:5: error: cannot find symbol
throw new IOException();
^
symbol: class IOException
location: class Example
1 error
G:\Java Programs>javac Example.java
Example.java:7: error: unreachable statement
System.out.println("After Exception");
^
Example.java:6: error: unreported exception IOException; must be caught or dec
red to be thrown
throw new IOException();
^
2 errors
G:\Java Programs>_

```
import java.io.*;
public class Example
{
 public static void main(String []args) throws IOException
 {
  throw new IOException();
//System.out.println("After Exception");
 }
}
```

```
import java.io.*;
public class Example
{
 public static void main(String []args)
 {

  try
  {
   throw new IOException();
  //System.out.println("After Exception");
  }
  catch(IOException e)
  { System.out.println("Exception:"+e.getMessage());}
 }
}
```

throws

- The throws keyword in java programming language is applicable to a method to indicate that the method raises particular type of exception while being processed.
- The throws keyword in java programming language takes arguments as a list of the objects of type java.lang.Throwables class.