

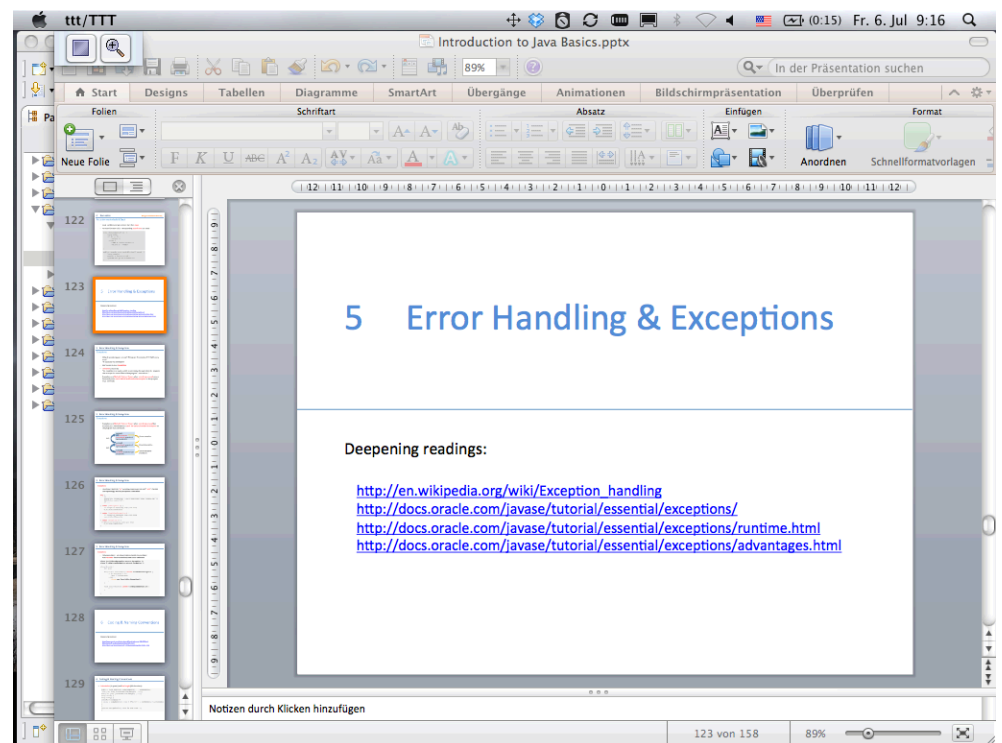
Script generated by TTT

Title: Lehmann: Uebung_Einf_HF (06.07.2012)

Date: Fri Jul 06 09:16:59 CEST 2012

Duration: 77:26 min

Pages: 84



5 Handling & Exceptions

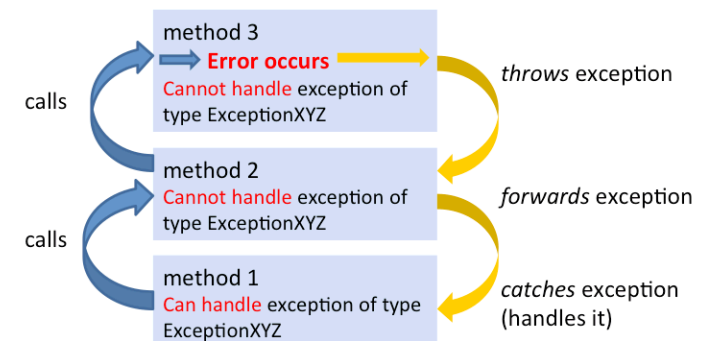
Exceptions

- What if something goes wrong? → Program Termination?!?! With every Error?
→ Obviously not intelligent!
- Mechanism in Java: **Exceptions**
- **Definition** [JTutorial]:
"An *exception* is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions."
- Exceptions are **like balls that are thrown** when **something unusual** occurs. Somebody must **catch the ball and handle the exception** or the program must terminate.

5 Handling & Exceptions

Exceptions

- Exceptions are **like balls that are thrown** when **something unusual** (an error) occurs. Somebody must **catch the ball and handle the exception** or the program must terminate.



5 Handling & Exceptions

Exceptions

- Usual case: Methods "try" possibly dangerous code and "catch" (handle) correspondingly resulting exceptions themselves

```
try {
    // ...
    FileWriter fileWriter = new FileWriter("someFileName.txt");
    fileWriter.write('a');
    // ...
} catch (IOException e) {
    // Exception handling code goes here
    e.printStackTrace();
} catch (SomeOtherException e) {
    // Exception handling code goes here
    e.printStackTrace();
} catch (Exception e) {
    // Exception handling code goes here
    e.printStackTrace();
}
```

5 Handling & Exceptions

Exceptions

- Other possibility: Let others (callers) handle the problem!
Add **throws** clause to method/constructor definition

```
class InvalidGearException extends Exception {}
class TireExplodedException extends Exception {}

class Bicycle {
    int gear;

    Bicycle(int initialGear) throws InvalidGearException {
        if (initialGear > 0) {
            gear = initialGear;
        } else {
            throw new InvalidGearException();
        }
    }

    void inflateTires() throws TireExplodedException {
        // ...
    }
}
```

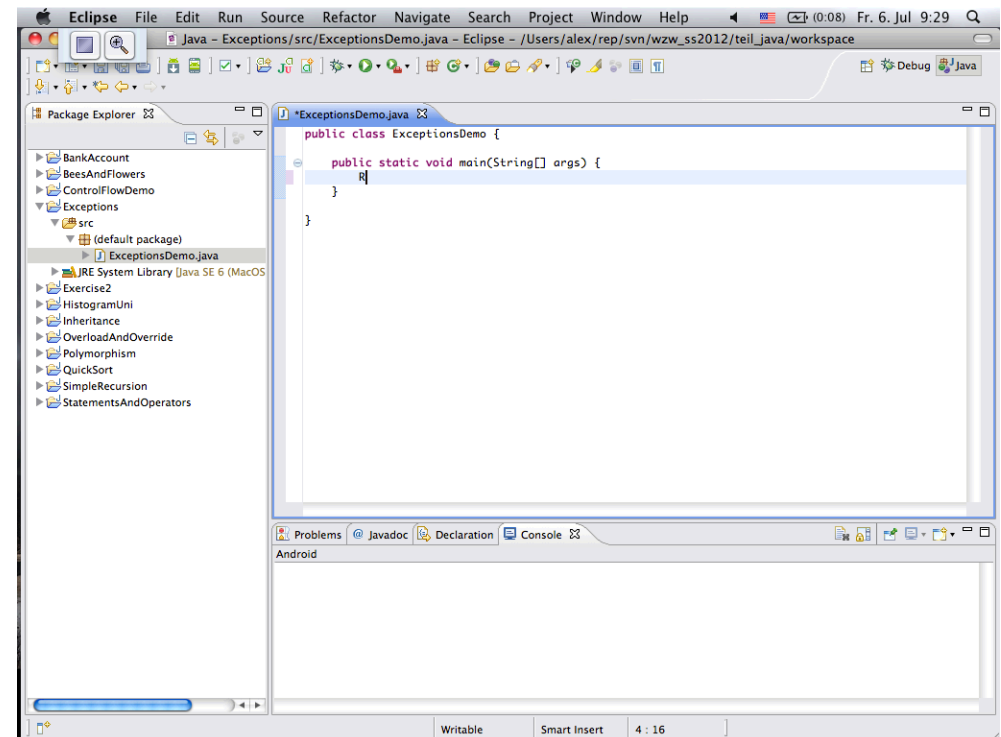


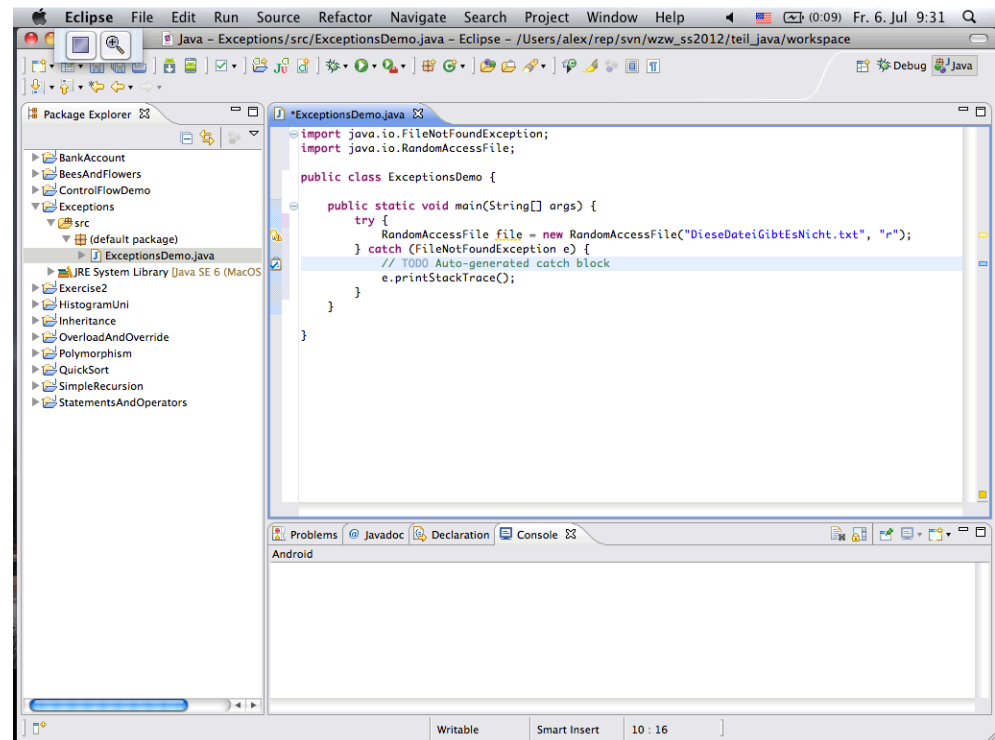
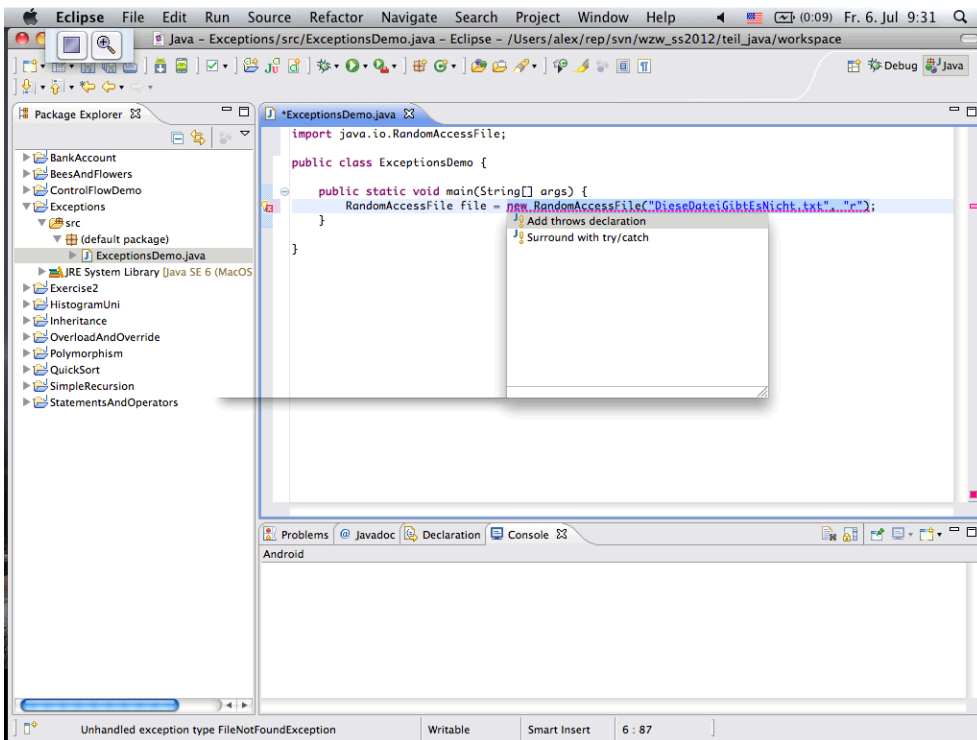
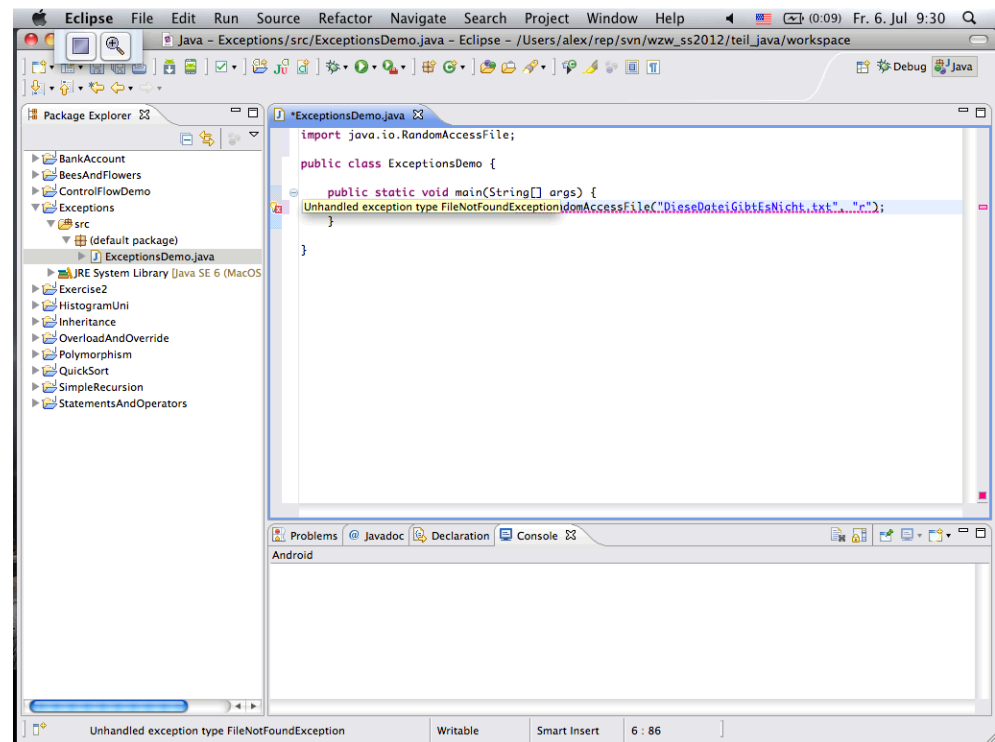
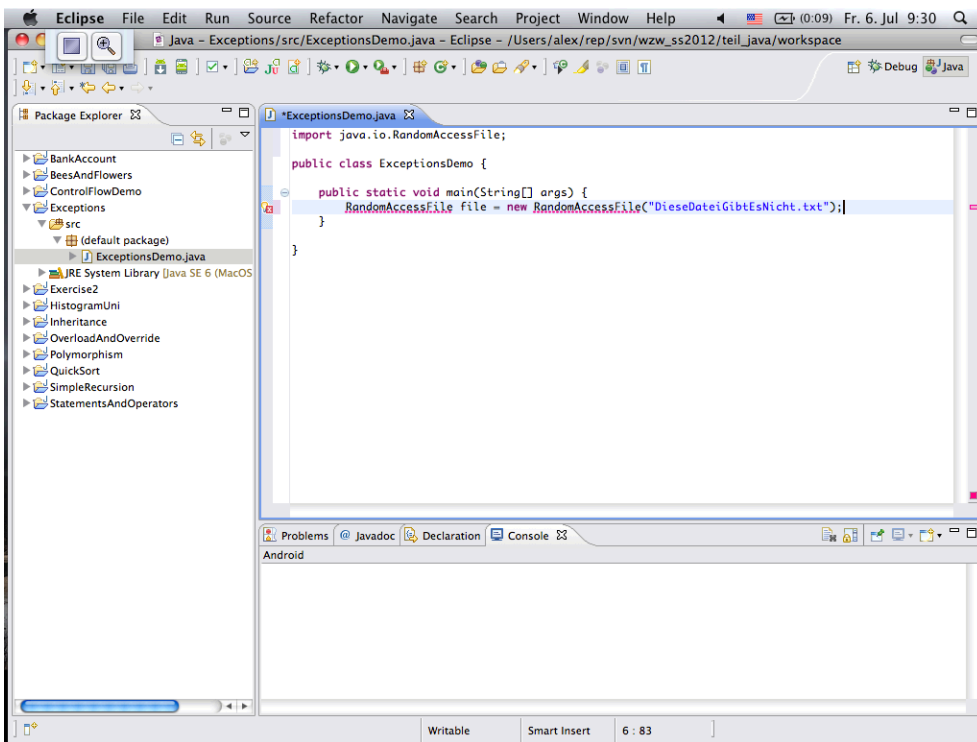
6 Coding & Naming Conventions

Deepening readings:

```
    if (initialGear > 0) {
        gear = initialGear;
    } else {
        throw new InvalidGearException();
    }
}

void inflateTires() throws TireExplodedException {
    // ...
}
}
```





Eclipse IDE screenshot showing the initial state of the ExceptionsDemo.java file. The code defines a main method that attempts to open a file named "DieseDateiGibtEsNicht.txt".

```

import java.io.FileNotFoundException;
import java.io.RandomAccessFile;

public class ExceptionsDemo {

    public static void main(String[] args) {
        try {
            RandomAccessFile file = new RandomAccessFile("DieseDateiGibtEsNicht.txt", "r");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
            System.out.println("Hier ist unsere Fehlerbehandlung.");
        }
    }
}

```

The console output shows a `java.io.FileNotFoundException: DieseDateiGibtEsNicht.txt (No such file or directory)` and the program prints "Hier ist unsere Fehlerbehandlung."

Eclipse IDE screenshot showing the first modification to ExceptionsDemo.java. A `catch` block for `FileNotFoundException` is added to the try-catch block.

```

import java.io.FileNotFoundException;
import java.io.RandomAccessFile;

public class ExceptionsDemo {

    public static void main(String[] args) {
        try {

        } catch (FileNotFoundException e) {
            e.printStackTrace();
            System.out.println("Hier ist unsere Fehlerbehandlung.");
        }
    }
}

```

The console output remains the same as in the first screenshot, showing the file not found error and the custom error message.

Eclipse IDE screenshot showing the second modification to ExceptionsDemo.java. The code is updated to handle `IOException` in addition to `FileNotFoundException`.

```

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;

public class ExceptionsDemo {

    public static void main(String[] args) {
        try {
            RandomAccessFile file = new RandomAccessFile("DieseDateiGibtEsNicht.txt", "r");
            String line = file.readLine();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
            System.out.println("Hier ist unsere Fehlerbehandlung.");
        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Hier ist unsere spezielle Fehlerbehandlung fuer IOException.");
        }
    }
}

```

The console output shows the `FileNotFoundException` error, followed by the `IOException` error, and the program prints "Hier ist unsere Fehlerbehandlung."

Eclipse IDE screenshot showing the final modification to ExceptionsDemo.java. The code is updated to handle `IOException` in addition to `FileNotFoundException` and `FileNotFoundException`.

```

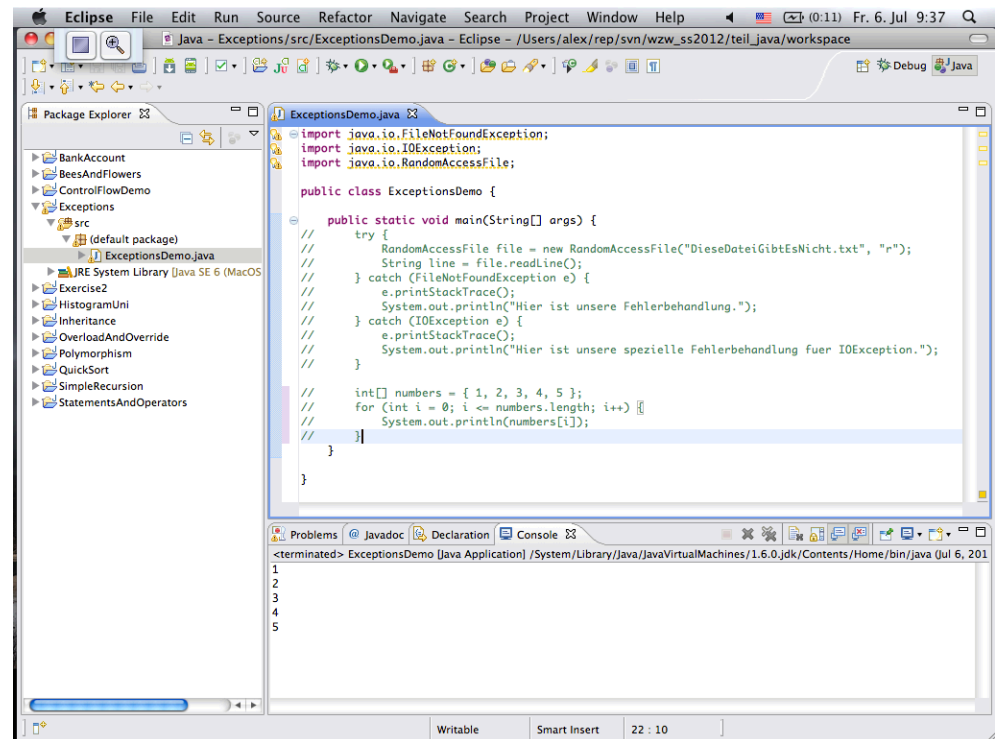
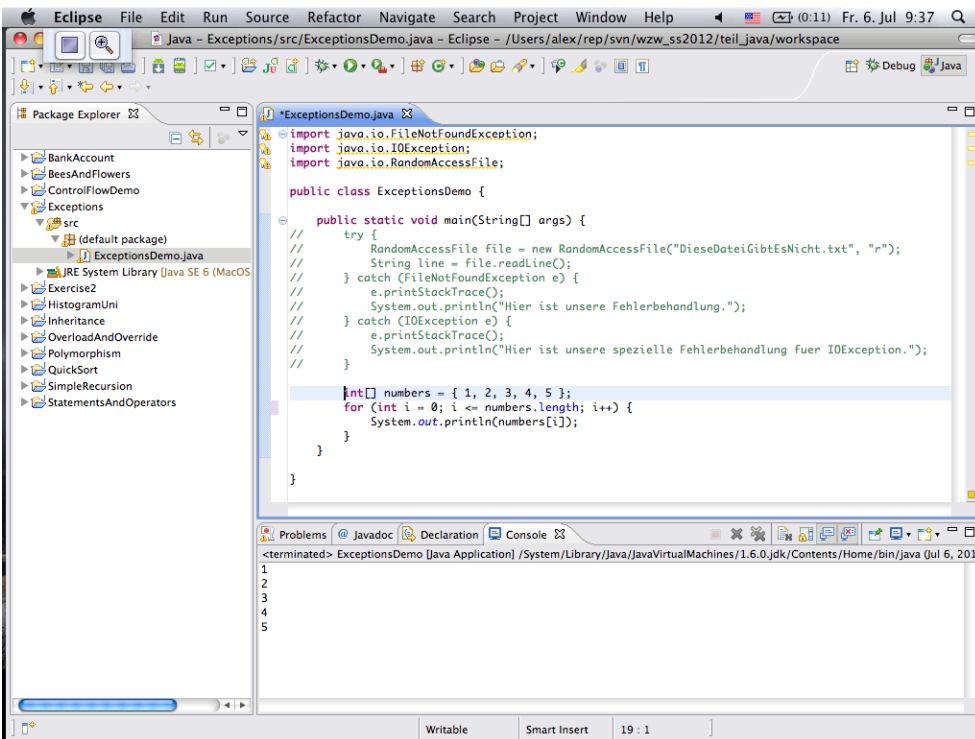
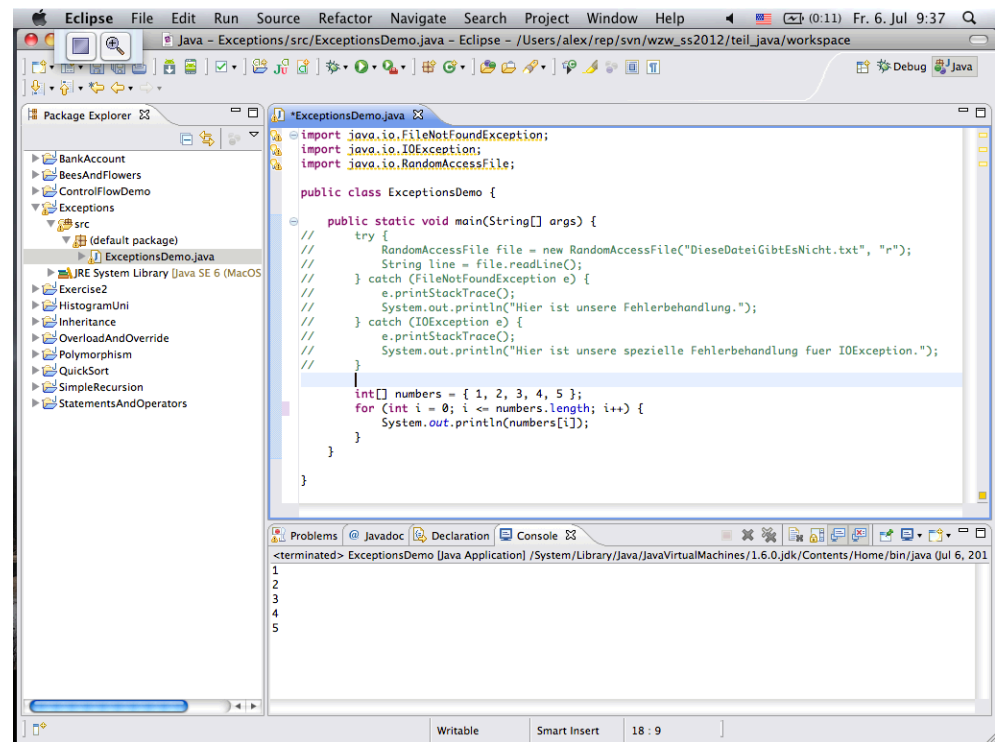
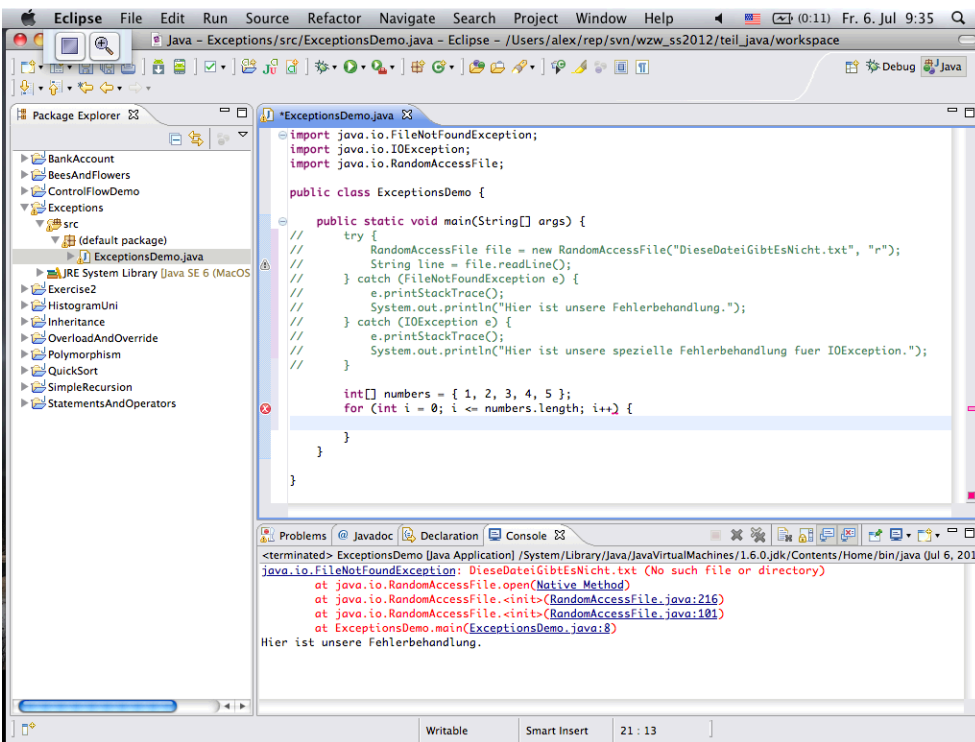
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;

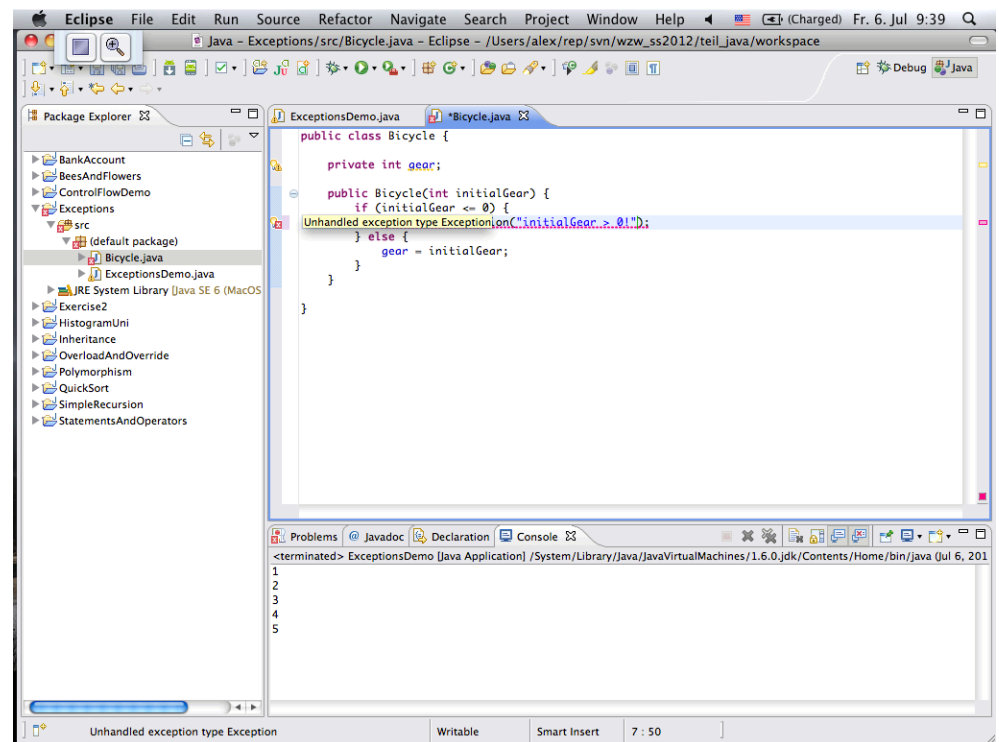
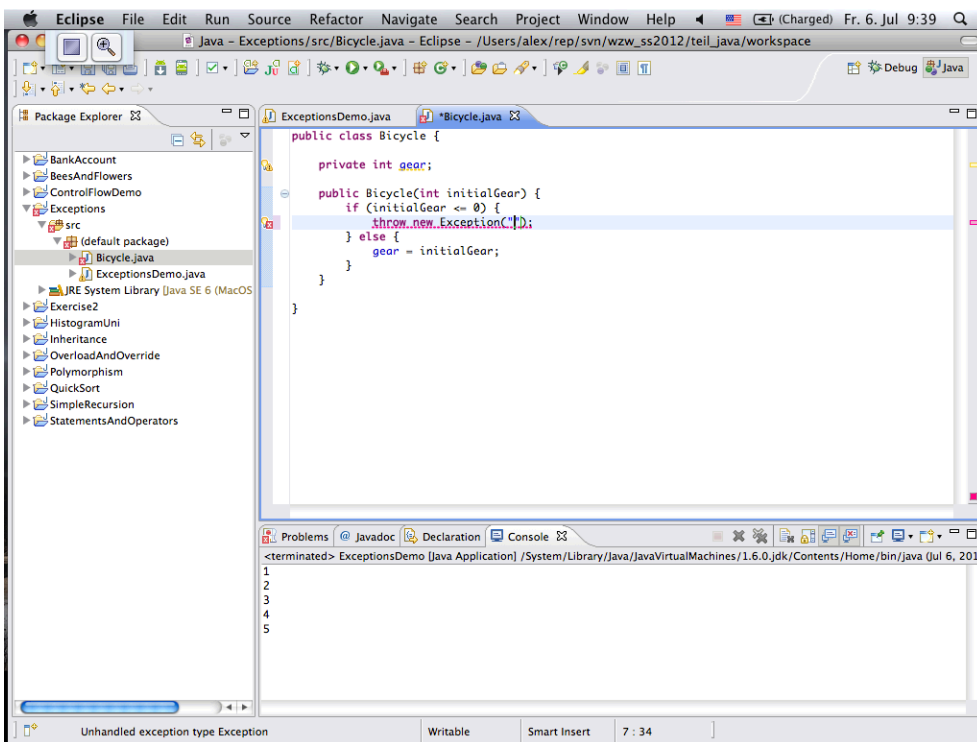
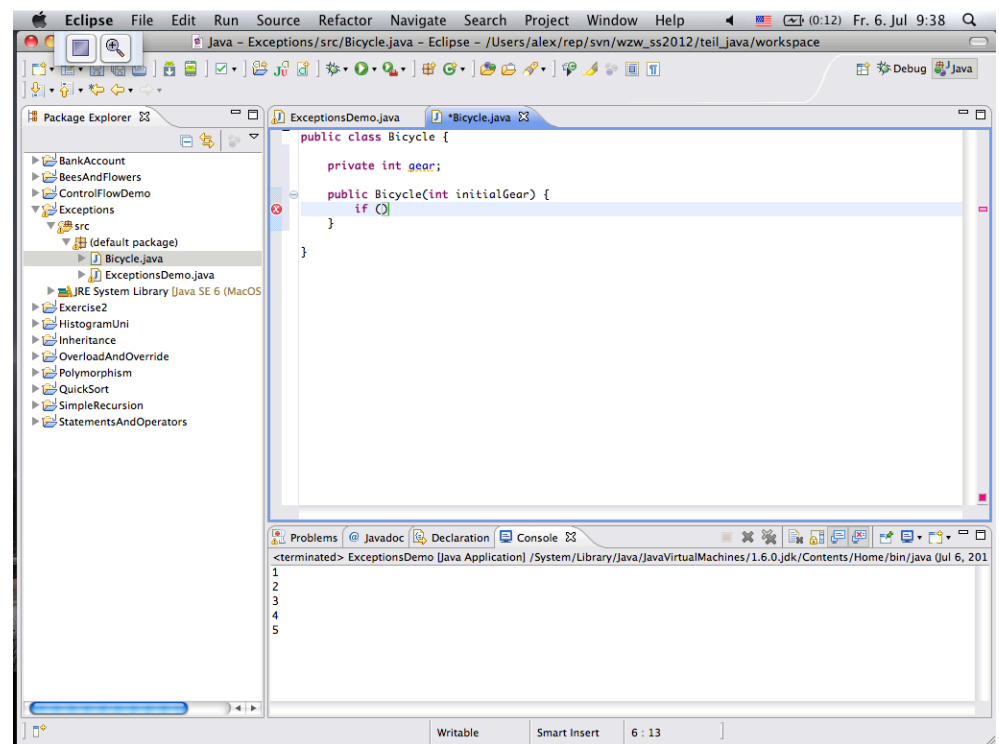
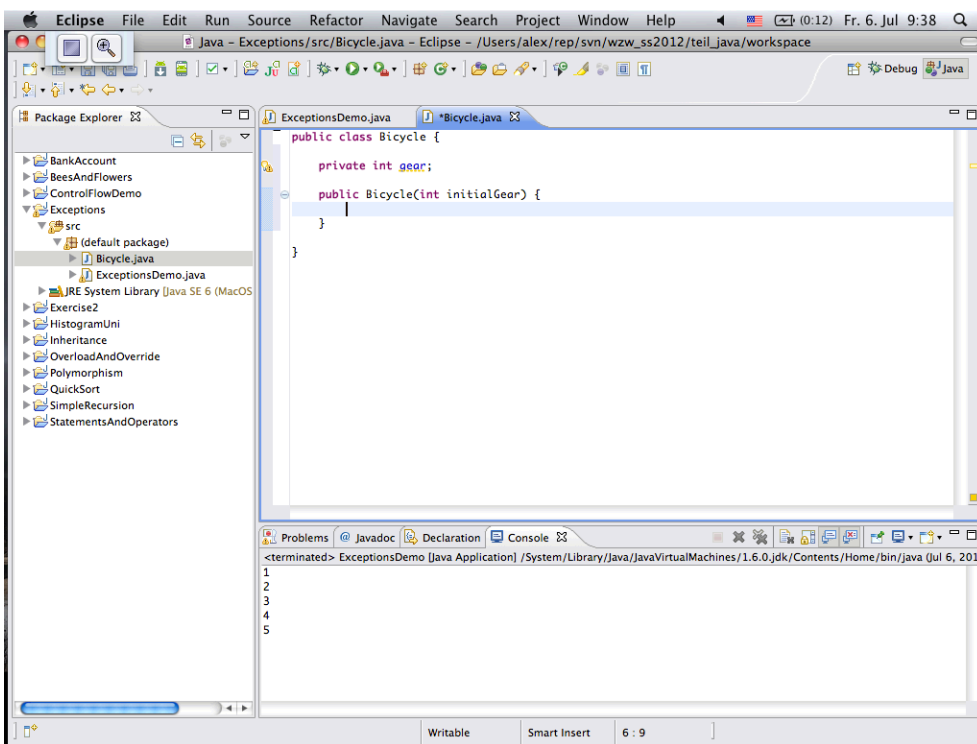
public class ExceptionsDemo {

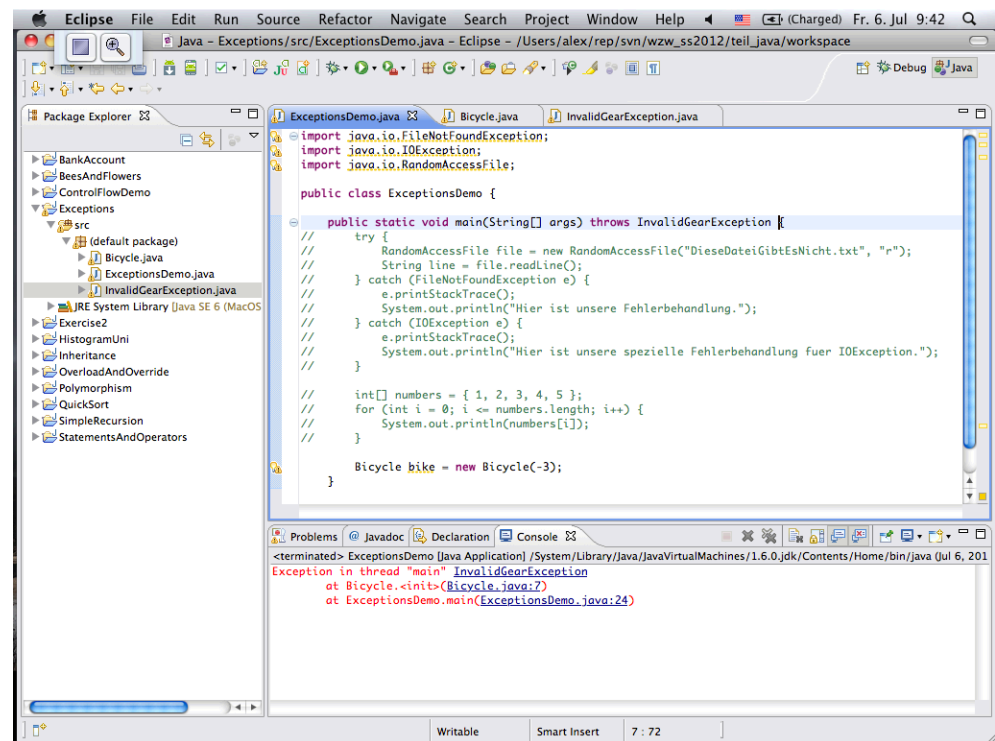
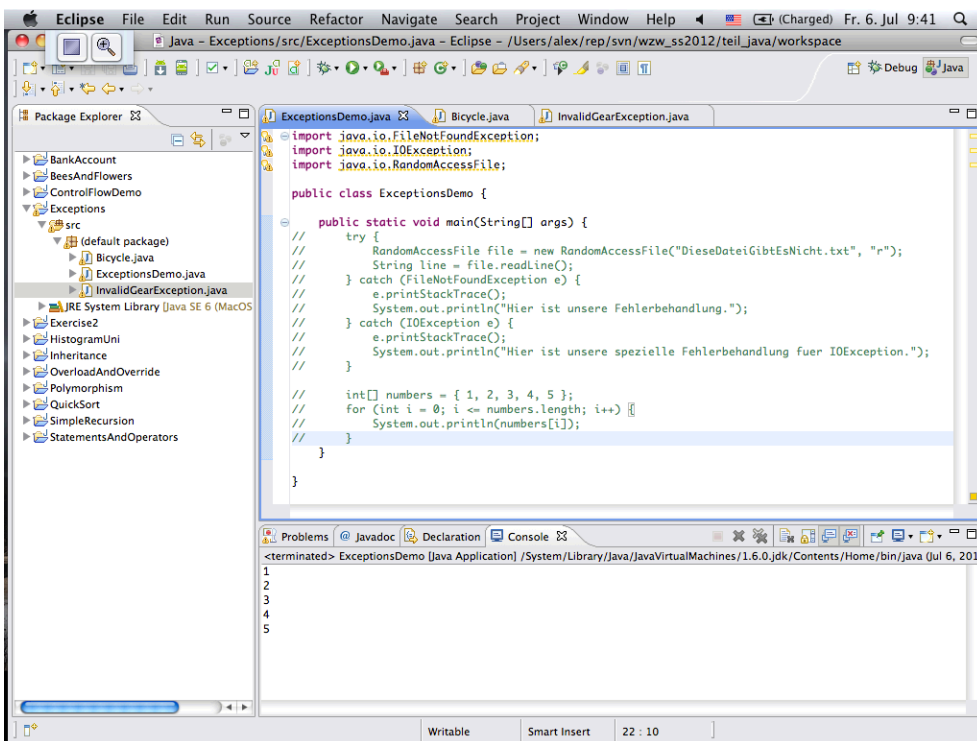
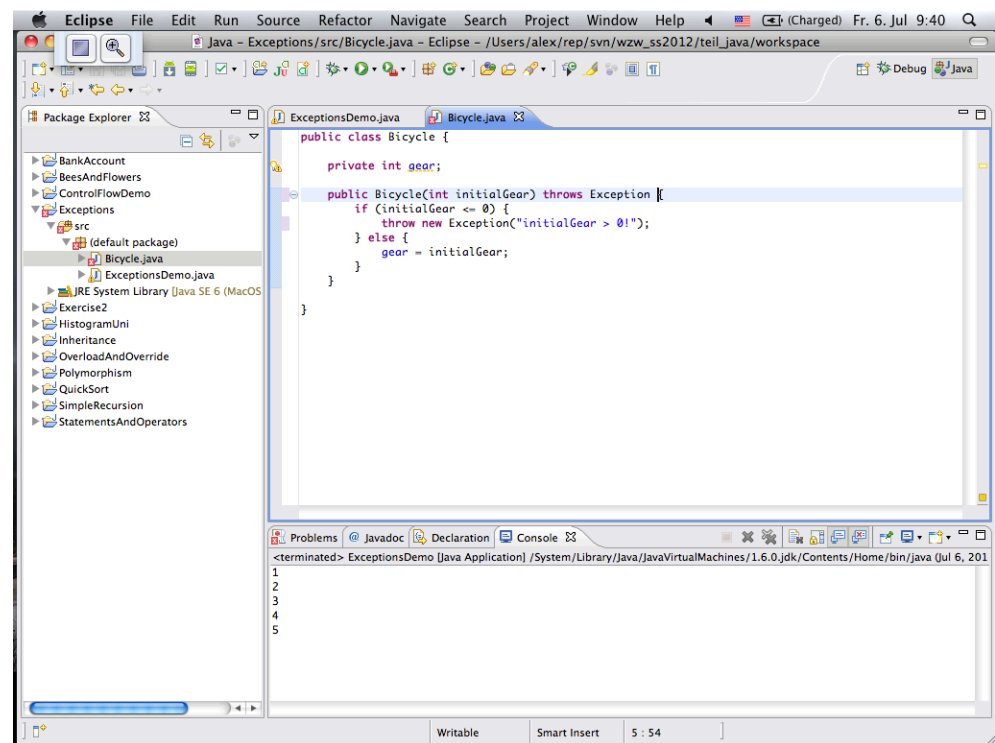
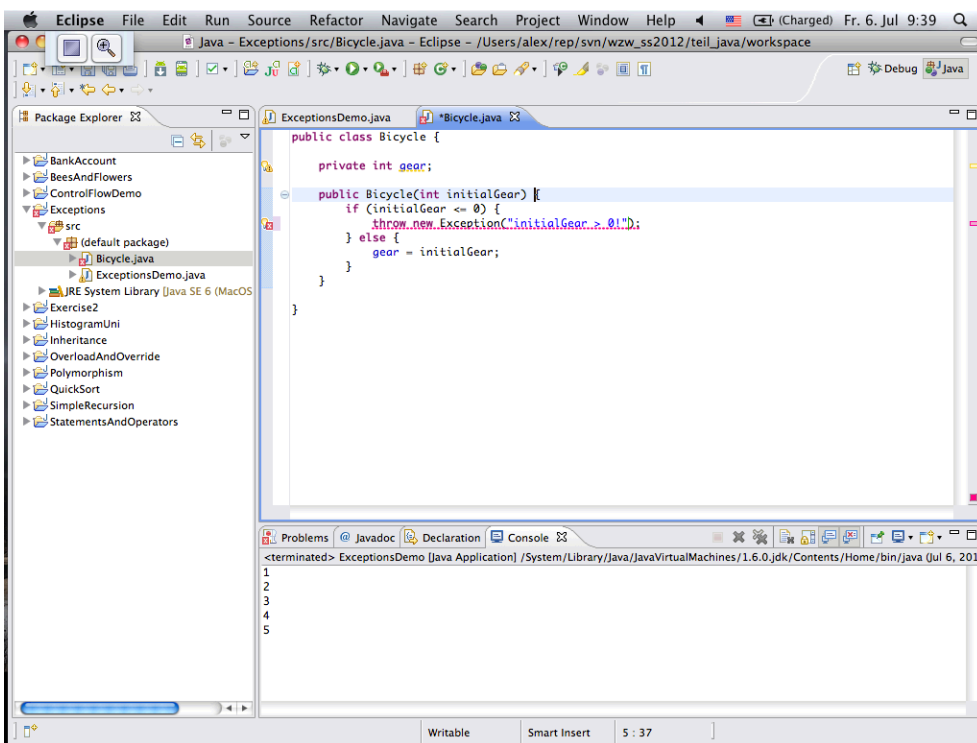
    public static void main(String[] args) {
        try {
            RandomAccessFile file = new RandomAccessFile("DieseDateiGibtEsNicht.txt", "r");
            String line = file.readLine();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
            System.out.println("Hier ist unsere Fehlerbehandlung.");
        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Hier ist unsere spezielle Fehlerbehandlung fuer IOException.");
        }
    }
}

```

The console output shows the `FileNotFoundException` error, followed by the `IOException` error, and the program prints "Hier ist unsere Fehlerbehandlung."









6 Coding & Naming Conventions

Deepening readings:

<http://www.oracle.com/technetwork/java/codeconv-138413.html>
<http://geosoft.no/development/javastyle.html>
<http://docs.oracle.com/javase/1.5.0/docs/guide/javadoc/index.html>



6 Coding & Naming Conventions

Deepening readings:

<http://www.oracle.com/technetwork/java/codeconv-138413.html>
<http://geosoft.no/development/javastyle.html>
<http://docs.oracle.com/javase/1.5.0/docs/guide/javadoc/index.html>

6 Coding & Naming Conventions

- **Indentation** (4 spaces) and **line length** (80 characters):

```
public void initializeMatrix(int[][] someMatrix) {
    for(int i=0; i<someMatrix.length; i++){
    for(int j=0; j<someMatrix.length; j++){
    if(i%2==0) {
    if(i==2*j) {
    someMatrix[i][j]=7;
    } else { someMatrix[i][j] = 3*i+72*j + (int)Math.floor(i/(j+1
    )
    }
    System.out.println("This is the end!");
    }
```

6 Coding & Naming Conventions

- **Indentation** (4 spaces) and **line length** (80 characters):

```
public void initializeMatrix(int[][] someMatrix) {
    for (int i = 0; i < someMatrix.length; i++) {
    for (int j = 0; j < someMatrix.length; j++) {
    if (i % 2 == 0) {
    if (i == 2 * j) {
    someMatrix[i][j] = 7;
    } else {
    someMatrix[i][j] = 3 * i + 72 * j +
    (int)Math.floor(i / (j + 1));
    }
    }
    }
    }
    System.out.println("This is the end!");
}
```


6 Coding & Naming Conventions

- **Indentation** (4 spaces) and **line length** (80 characters):

```
public void initializeMatrix(int[][] someMatrix) {
    for (int i = 0; i < someMatrix.length; i++) {
        for (int j = 0; j < someMatrix.length; j++) {
            if (i % 2 == 0) {
                if (i == 2 * j) {
                    someMatrix[i][j] = 7;
                } else {
                    someMatrix[i][j] = 3 * i + 72 * j +
                        (int)Math.floor(i / (j + 1));
                }
            }
        }
    }
    System.out.println("This is the end!");
}
```

Tip: Use IDE/editor with syntax highlighting!



6 Coding & Naming Conventions

- **Names of variables and methods**

- **Short and meaningful**
- *Indicate to the casual observer the intent of its use*
- *Mixed case with the first letter lowercase, with the first letter of each internal word capitalized*
- *Booleans with prefix "is"*
- **Examples:**

```
int rowIndex
int columnIndex
boolean isValid
static boolean copyFile(URL from, URL to)
```

- **Names of constants**

- *Uppercase with words separated by underscores*

```
static final int MIN_ALLOWED_VALUE = 42;
```

6 Coding & Naming Conventions

- **Names of classes and interfaces**

- **Short and meaningful**
- *Nouns in mixed case with the first letter lowercase, with the first letter of each internal word capitalized*
- *Good style: Use only [a..z] and [A..Z]*
- **Examples:**

```
class Bicycle
class ImageSprite
interface RasterDelegate
interface Comparable
```

6 Coding & Naming Conventions

- Use **English for everything**, e.g. names, comments etc.
- **Adhere STRICTLY** to *common* coding styles
➔ Do **not use exotic** "own" coding styles!
- **Use default style of IDE**
(Integrated Development Environment, e.g. Eclipse)
- Use **comments REGULARLY and THOROUGHLY**
- Use **Javadoc** (not only for larger projects)



7 Using the Java Class Library

Deepening readings (optional):

<http://docs.oracle.com/javase/tutorial/java/data/index.html>
<http://docs.oracle.com/javase/tutorial/java/generics/index.html>

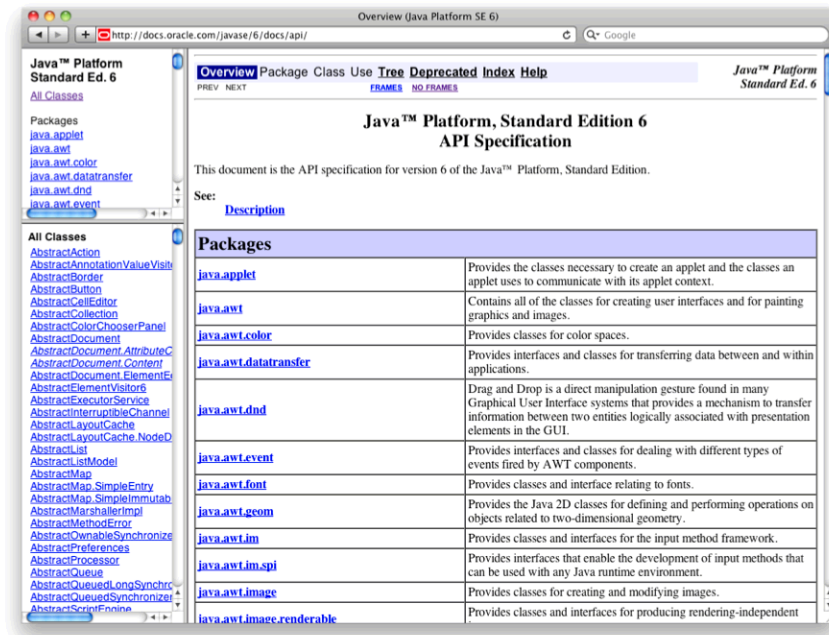
Main reference:

<http://docs.oracle.com/javase/6/docs/api/>

7 Using the Java Class Library

- **So far:**
 - Classes, objects, methods, fields, variables, control flow statements etc.
 - These are only *basic tools*
- **Advanced problems** require a **multitude** of **specialized types** (classes etc.)
- **Write** them all by **yourself**?
 - **No!** Java's strength: **Types** for almost EVERYTHING exist
- **"Class Library"**
 - Documentation at <http://java.sun.com/javase/6/docs/api/>
- Similar class libraries exist for .NET, C++, and other languages/platforms

7 Using the Java Class Library



7 Using the Java Class Library

Packages and Imports

- Java's classes and types are organized in **hierarchical packages**

```
java.lang.String
java.net.URLConnection
java.util.Collection
javax.xml.parsers.SAXParser
org.w3c.dom.events.DocumentEvent
```

- All types from package `java.lang` are imported automatically
- Other types need to be **imported**

```
import java.net.URLConnection;
import java.util.Collection;
import some.other.package.*; // * means all types
```

7 Navigating the Java Class Library



7 Navigating the Java Class Library

Packages and Imports

- Java's classes and types are organized in **hierarchical packages**

```
java.lang.String
java.net.URLConnection
java.util.Collection
javax.xml.parsers.SAXParser
org.w3c.dom.events.DocumentEvent
```

- All types from package `java.lang` are imported automatically
- Other types need to be **imported**

```
import java.net.URLConnection;
import java.util.Collection;
import some.other.package.*; // * means all types
```

7 Navigating the Java Class Library

Generics

- Some types may be **parameterized with other types**
- Typical examples are classes that implement data structures
- Advantages:**
 - Type checks** at compile time
 - Programmers can **implement algorithms generically**
- Examples:**

```
Vector nonGenericVector = new Vector();
nonGenericVector.add( 1234 );
nonGenericVector.add( "Hello world" );
Object typeUnknown = nonGenericVector.get(0);
```

```
Vector<Bicycle> bicycles = new Vector<Bicycle>();
bicycles.add( new Bicycle() );
bicycles.add( 123 ); // Compile time error!
Bicycle typeKnown = bicycles.get(0);
```

7 Navigating the Java Class Library

Wrapper-Classes

- Parameters restricted to *reference types*** (classes, interfaces)
- For each *primitive type*, a corresponding **wrapper class** exists
- Examples:

```
java.lang.Short
java.lang.Integer
java.lang.Long
java.lang.Float
java.lang.Double
```

etc.

- Primitive types are automatically **boxed** and **unboxed** when necessary

```
Integer i = 723;
int j = i;
```



8 Solving a Problem

Deepening readings (optional):

[The Internet](#)

Main reference:

<http://docs.oracle.com/javase/6/docs/api/>

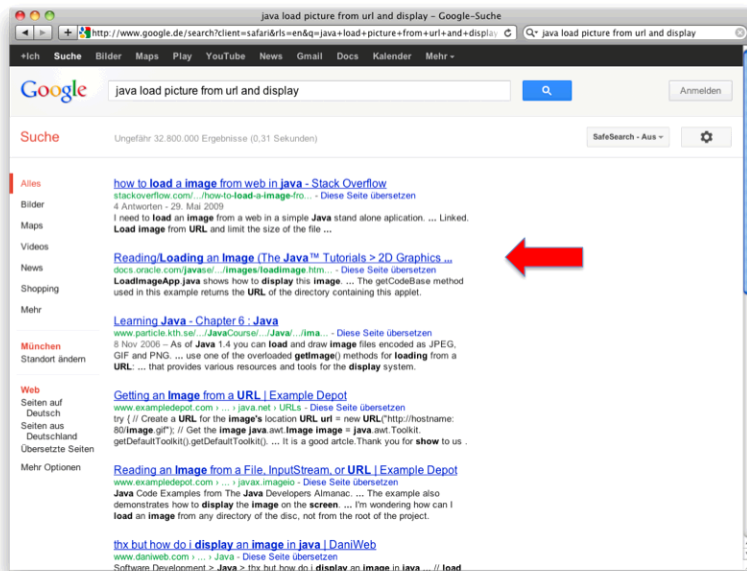
8 Solving a Problem

Example Task

- **Example task:**
Load an image from the internet and display that image
- How would **you** start?
 - **Google** is your friend
 - So is the **Java Tutorial**
 - Look up corresponding classes, fields and methods in **API documentation**
 - Write program
 - Test program
 - Repeat all of the above as long as necessary
 - DONE 😊

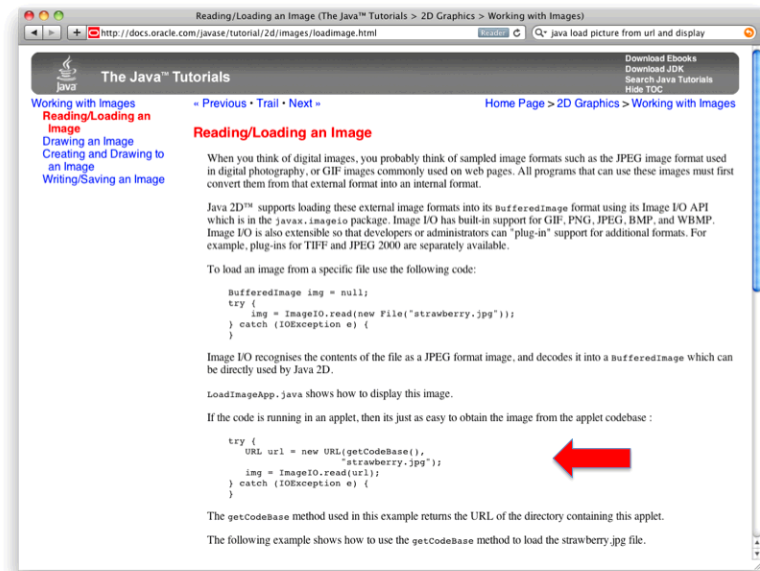
8 Solving a Problem

Example Task



8 Solving a Problem

Example Task



8 Solving a Problem

Example Task

Introduction to Java Basics.pptx

8 Solving a Problem

Example Task

The Java™ Tutorials

Reading/Loading an Image

When you think of digital images, you probably think of sampled image formats such as the JPEG image format used in digital photography, or GIF images commonly used on web pages. All programs that can use these images must first convert them from that external format into an internal format.

Java 2D™ supports loading these external image formats into its `BufferedImage` format using its Image IO API which is in the `javax.imageio` package. Image IO has built-in support for GIF, PNG, JPEG, BMP, and WBMP. Image IO is also extensible so that developers or administrators can "plug-in" support for additional formats. For example, plug-ins for TIFF and JPEG 2000 are separately available.

To load an image from a specific file use the following code:

```
BufferedImage img = null;
try {
    img = ImageIO.read(new File("strawberry.jpg"));
} catch (IOException e) {
}
```

Image IO recognises the contents of the file as a JPEG format image, and decodes it into a `BufferedImage` which can be directly used by Java 2D.

`LoadImageApp.java` shows how to display this image.

If the code is running in an applet, then it's just as easy to obtain the image from the applet codebase:

```
try {
    URL url = new URL(getCodeBase(),
        "strawberry.jpg");
    img = ImageIO.read(url);
} catch (IOException e) {
}
```

The `getCodeBase` method used in this example returns the URL of the directory containing this applet.

The following example shows how to use the `getCodeBase` method to load the `strawberry.jpg` file.

Reading/Loading an Image (The Java™ Tutorials > 2D Graphics > Working with Images)

The Java™ Tutorials

Working with Images

Reading/Loading an Image

Reading/Loading an Image

When you think of digital images, you probably think of sampled image formats such as the JPEG image format used in digital photography, or GIF images commonly used on web pages. All programs that can use these images must first convert them from that external format into an internal format.

Java 2D™ supports loading these external image formats into its `BufferedImage` format using its Image IO API which is in the `javax.imageio` package. Image IO has built-in support for GIF, PNG, JPEG, BMP, and WBMP. Image IO is also extensible so that developers or administrators can "plug-in" support for additional formats. For example, plug-ins for TIFF and JPEG 2000 are separately available.

To load an image from a specific file use the following code:

```
BufferedImage img = null;
try {
    img = ImageIO.read(new File("strawberry.jpg"));
} catch (IOException e) {
}
```

Image IO recognises the contents of the file as a JPEG format image, and decodes it into a `BufferedImage` which can be directly used by Java 2D.

`LoadImageApp.java` shows how to display this image.

If the code is running in an applet, then it's just as easy to obtain the image from the applet codebase:

```
try {
    URL url = new URL(getCodeBase(),
        "strawberry.jpg");
    img = ImageIO.read(url);
} catch (IOException e) {
}
```

The `getCodeBase` method used in this example returns the URL of the directory containing this applet.

The following example shows how to use the `getCodeBase` method to load the `strawberry.jpg` file.

8 Solving a Problem

Example Task

8 Solving a Problem

Example Task

Reading/Loading an Image (The Java™ Tutorials > 2D Graphics > Working with Images)

The Java™ Tutorials

Working with Images

Reading/Loading an Image

Reading/Loading an Image

When you think of digital images, you probably think of sampled image formats such as the JPEG image format used in digital photography, or GIF images commonly used on web pages. All programs that can use these images must first convert them from that external format into an internal format.

Java 2D™ supports loading these external image formats into its `BufferedImage` format using its Image IO API which is in the `javax.imageio` package. Image IO has built-in support for GIF, PNG, JPEG, BMP, and WBMP. Image IO is also extensible so that developers or administrators can "plug-in" support for additional formats. For example, plug-ins for TIFF and JPEG 2000 are separately available.

To load an image from a specific file use the following code:

```
BufferedImage img = null;
try {
    img = ImageIO.read(new File("strawberry.jpg"));
} catch (IOException e) {
}
```

Image IO recognises the contents of the file as a JPEG format image, and decodes it into a `BufferedImage` which can be directly used by Java 2D.

`LoadImageApp.java` shows how to display this image.

If the code is running in an applet, then it's just as easy to obtain the image from the applet codebase:

```
try {
    URL url = new URL(getCodeBase(),
        "strawberry.jpg");
    img = ImageIO.read(url);
} catch (IOException e) {
}
```

The `getCodeBase` method used in this example returns the URL of the directory containing this applet.

The following example shows how to use the `getCodeBase` method to load the `strawberry.jpg` file.

URL (Java Platform SE 6)

Overview Package Class Use Tree Deprecated Index Help

Class URL

`java.lang.Object`
`java.net.URL`

All Implemented Interfaces:

`Serializable`

```
public final class URL
extends Object
implements Serializable
```

Class `URL` represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web. A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine. More information on the types of URLs and their formats can be found at:

<http://www.socs.uts.edu.au/MosaicDocs-old/url-primer.html>

In general, a URL can be broken into several parts. The previous example of a URL indicates that the protocol to use is `http` (Hypertext Transfer Protocol) and that the information resides on a host machine named `www.socs.uts.edu.au`. The information on that host machine is named `MosaicDocs-old/url-primer.html`. The exact meaning of this name on the host machine is both protocol dependent and host dependent. The information normally resides in a file, but it could be generated on the fly. This component of the URL is called the *path* component.

A URL can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for `http` is 80. An alternative port could be specified as:

```
http://www.socs.uts.edu.au:80/MosaicDocs-old/url-primer.html
```

The syntax of `url` is defined by [RFC 2396: Uniform Resource Identifiers \(URI\): Generic Syntax](#), amended by [RFC 2732: Format for Literal IPv6 Addresses in URIs](#). The Literal IPv6 address format also supports `scope_ids`. The syntax and usage of `scope_ids` is described [here](#).

A URL may have appended to it a "fragment", also known as a "ref" or a "reference". The fragment is indicated by the sharp sign character `#` followed by more characters. For example,

8 Solving a Problem

Example Task

URL (Java Platform SE 6)

http://docs.oracle.com/javase/6/docs/api/java/net/URL.html

Note: the `URL` class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use `URLEncoder`, and to convert between these two classes using `URLEncoder` and `URLDecoder`.

The `URLEncoder` and `URLDecoder` classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

Since:
JDK1.0

See Also:
[Serialized Form](#)

Constructor Summary

<code>URL(String spec)</code>	Creates a <code>URL</code> object from the <code>String</code> representation.
<code>URL(String protocol, String host, int port, String file)</code>	Creates a <code>URL</code> object from the specified protocol, host, port number, and file.
<code>URL(String protocol, String host, int port, String file, URLStreamHandler handler)</code>	Creates a <code>URL</code> object from the specified protocol, host, port number, file, and handler.
<code>URL(String protocol, String host, String file)</code>	Creates a <code>URL</code> from the specified protocol name, host name, and file name.
<code>URL(URLContext context, String spec)</code>	Creates a <code>URL</code> by parsing the given spec within a specified context.
<code>URL(URLContext context, String spec, URLStreamHandler handler)</code>	Creates a <code>URL</code> by parsing the given spec with the specified handler within a specified context.

Method Summary

<code>boolean equals(Object obj)</code>	Compares this <code>URL</code> for equality with another object.
<code>String getAuthority()</code>	Gets the authority part of this <code>URL</code> .
<code>Object getContents()</code>	Gets the contents of this <code>URL</code> .

8 Solving a Problem

Example Task

URL (Java Platform SE 6)

http://docs.oracle.com/javase/6/docs/api/java/net/URL.html#URL(java.lang.String)

URL

```
public URL(String spec)
    throws MalformedURLException
```

Creates a `URL` object from the `String` representation.

This constructor is equivalent to a call to the two-argument constructor with a `null` first argument.

Parameters:
`spec` - the `String` to parse as a `URL`.

Throws:
`MalformedURLException` - If the string specifies an unknown protocol.

See Also:
[URL\(java.net.URL, java.lang.String\)](#)

URL

```
public URL(URLContext context,
           String spec)
    throws MalformedURLException
```

Creates a `URL` by parsing the given spec within a specified context. The new `URL` is created from the given context `URL` and the spec argument as described in RFC2396 "Uniform Resource Identifiers - Generic Syntax":

```
<scheme>://<authority><path>?<query>#<fragment>
```

The reference is parsed into the scheme, authority, path, query and fragment parts. If the path component is empty and the scheme, authority, and query components are undefined, then the new `URL` is a reference to the current document. Otherwise, the fragment and query parts present in the spec are used in the new `URL`.

If the scheme component is defined in the given spec and does not match the scheme of the context, then the new `URL` is created as an absolute `URL` based on the spec alone. Otherwise the scheme component is inherited from the context `URL`.

If the authority component is present in the spec then the spec is treated as absolute and the spec authority and path will replace the context authority and path. If the authority component is absent in the spec then the authority of the new `URL` will be inherited from the context.

If the spec's path component begins with a slash character "/" then the path is treated as absolute and the spec path replaces the context path.

Otherwise, the path is treated as a relative path and is appended to the context path, as described in RFC2396. Also, in this case, the path is canonicalized through the removal of directory changes made by occurrences of "." and "..".

Introduction to Java Basics.pptx

89%

8 Solving a Problem

Example Task

URL (Java Platform SE 6)

http://docs.oracle.com/javase/6/docs/api/java/net/URL.html

Note: the `URL` class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use `URLEncoder`, and to convert between these two classes using `URLEncoder` and `URLDecoder`.

The `URLEncoder` and `URLDecoder` classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

Since:
JDK1.0

See Also:
[Serialized Form](#)

Constructor Summary

<code>URL(String spec)</code>	Creates a <code>URL</code> object from the <code>String</code> representation.
<code>URL(String protocol, String host, int port, String file)</code>	Creates a <code>URL</code> object from the specified protocol, host, port number, and file.
<code>URL(String protocol, String host, int port, String file, URLStreamHandler handler)</code>	Creates a <code>URL</code> object from the specified protocol, host, port number, file, and handler.
<code>URL(String protocol, String host, String file)</code>	Creates a <code>URL</code> from the specified protocol name, host name, and file name.
<code>URL(URLContext context, String spec)</code>	Creates a <code>URL</code> by parsing the given spec within a specified context.
<code>URL(URLContext context, String spec, URLStreamHandler handler)</code>	Creates a <code>URL</code> by parsing the given spec with the specified handler within a specified context.

Method Summary

<code>boolean equals(Object obj)</code>	Compares this <code>URL</code> for equality with another object.
<code>String getAuthority()</code>	Gets the authority part of this <code>URL</code> .
<code>Object getContents()</code>	Gets the contents of this <code>URL</code> .

Notizen durch Klicken hinzufügen

Normalansicht 147 von 158 89%

Eclipse File Edit Run Source Navigate Search Project Refactor Window Help

Java - ImageDemo/src/ImageDemo.java - Eclipse - /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace

```
public class ImageDemo {
    public static void main(String[] args) {
        URL url = new URL("http://vmschlichter7.informatik.tu-muenchen.de/wzw_ss12/morningDew.jpg");
    }
}
```

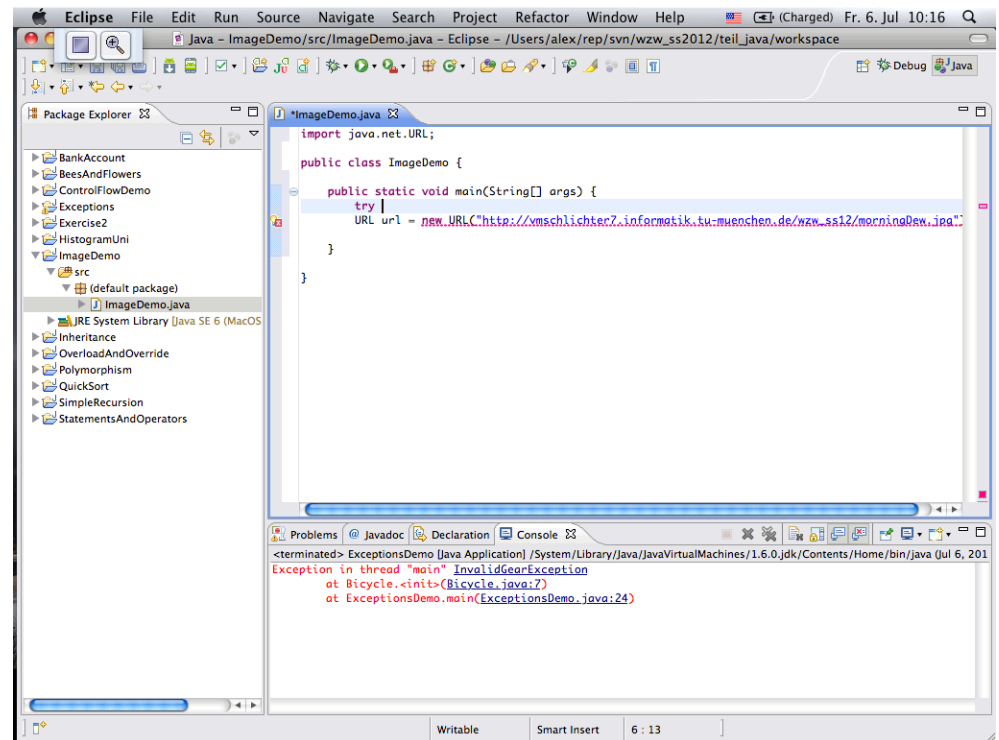
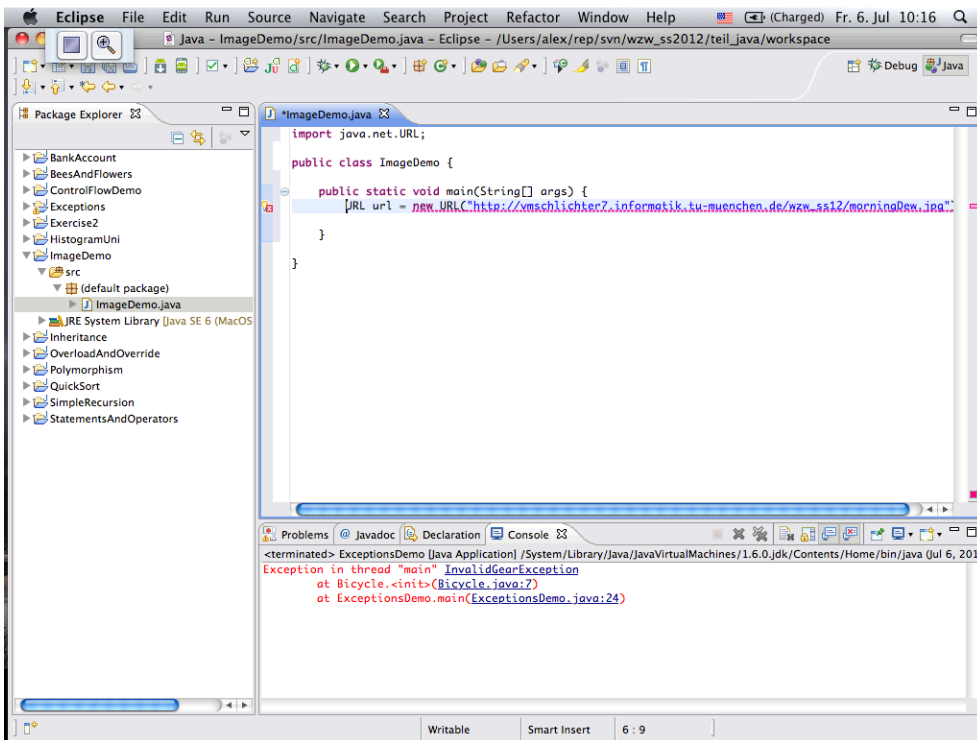
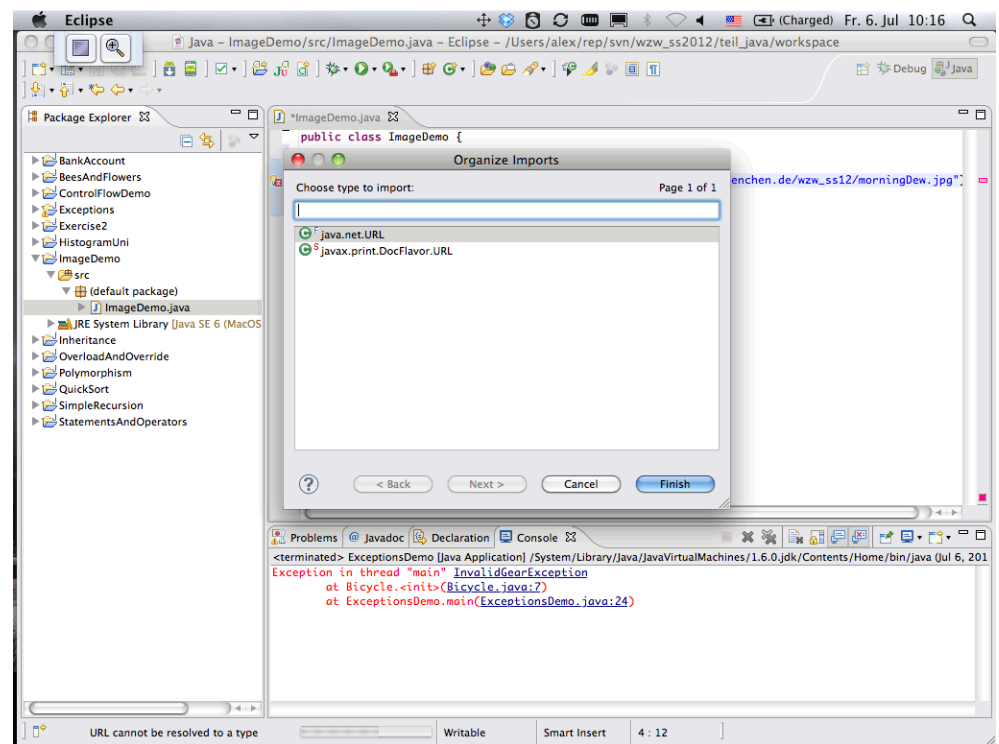
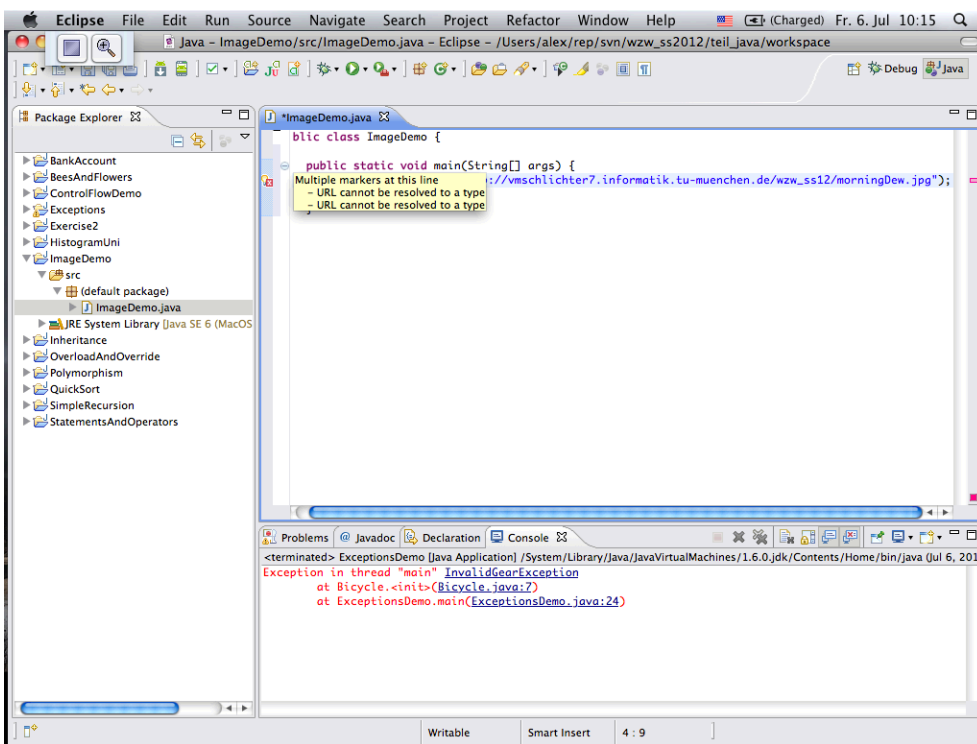
Package Explorer

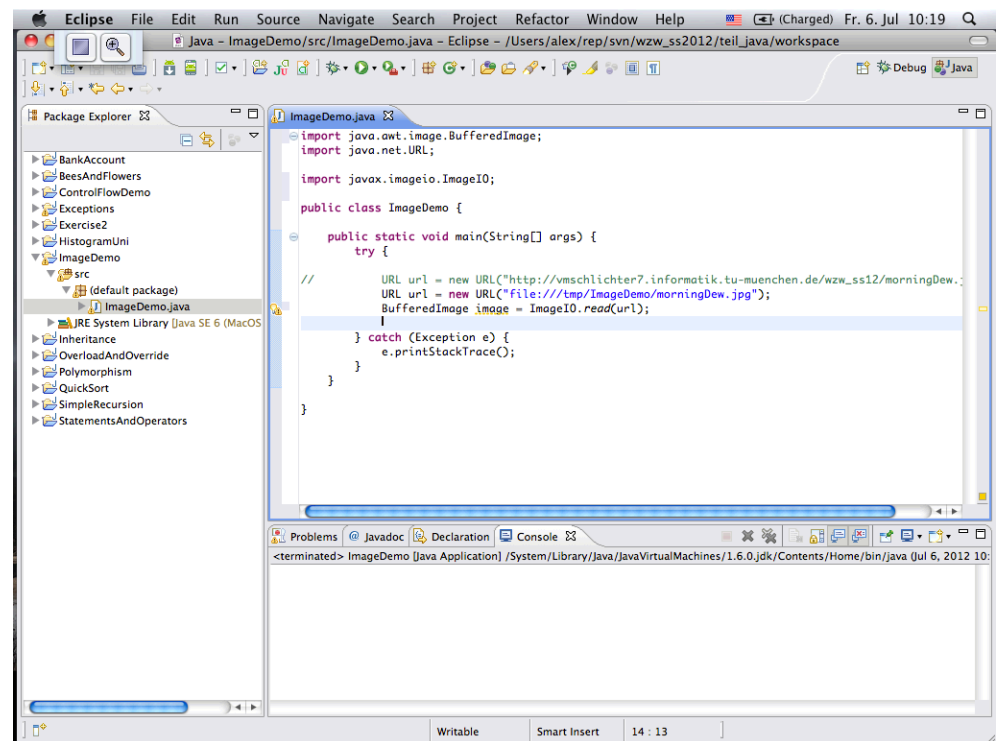
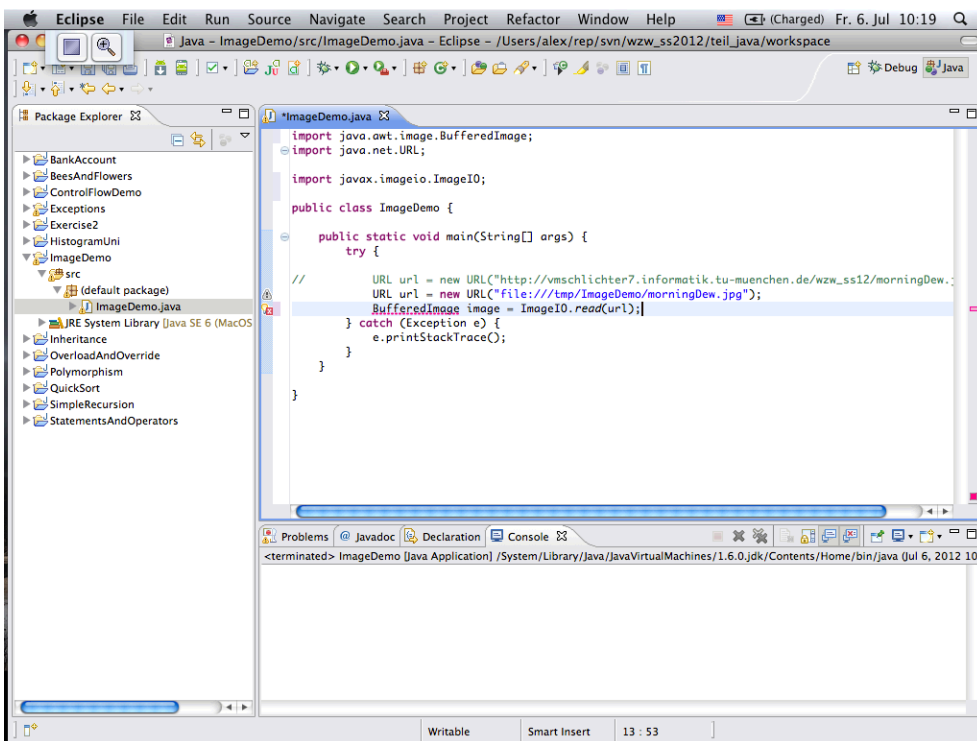
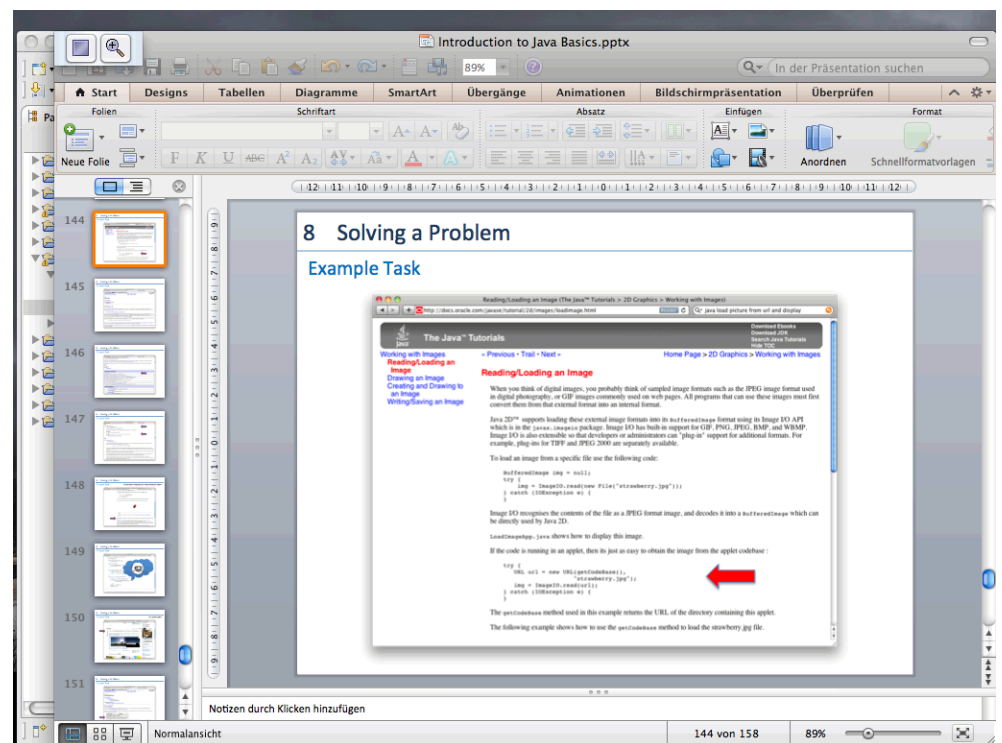
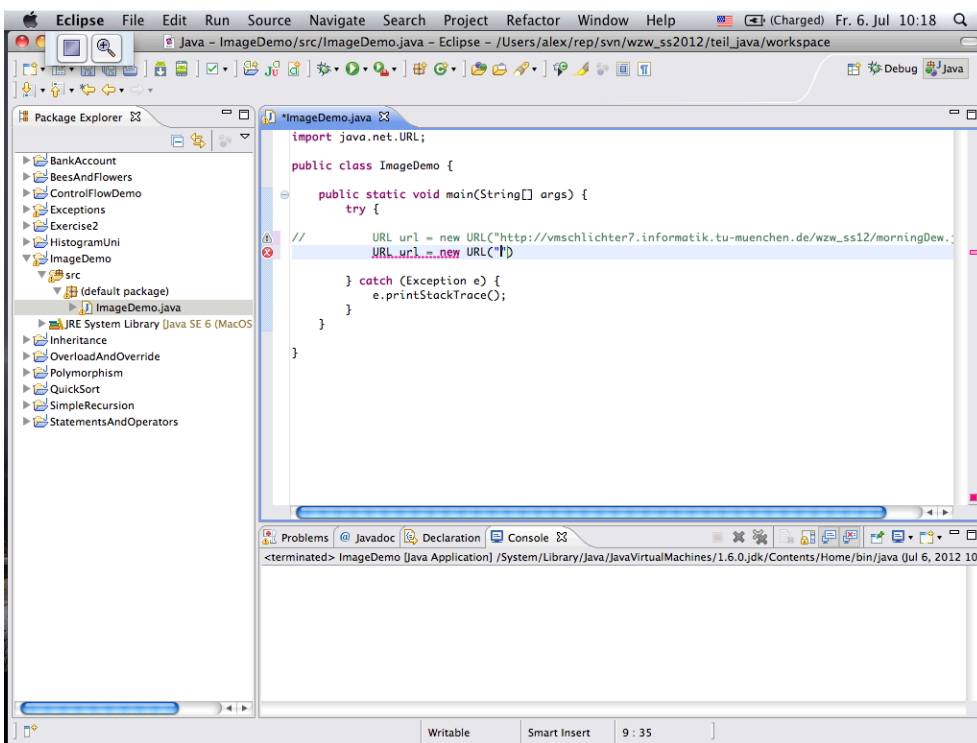
- BankAccount
- BeesAndFlowers
- ControlFlowDemo
- Exceptions
- Exercise2
- HistogramUni
- ImageDemo
- src
- (default package)
- ImageDemo.java
- JRE System Library [Java SE 6 (MacOS)
- Inheritance
- OverloadAndOverride
- Polymorphism
- QuickSort
- SimpleRecursion
- StatementsAndOperators

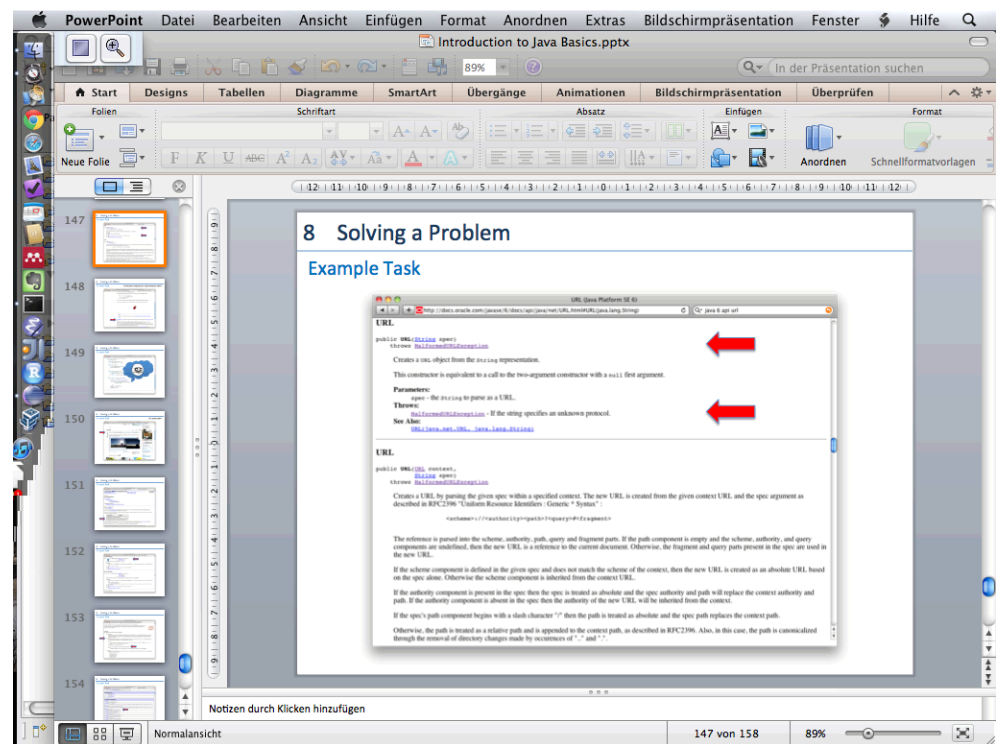
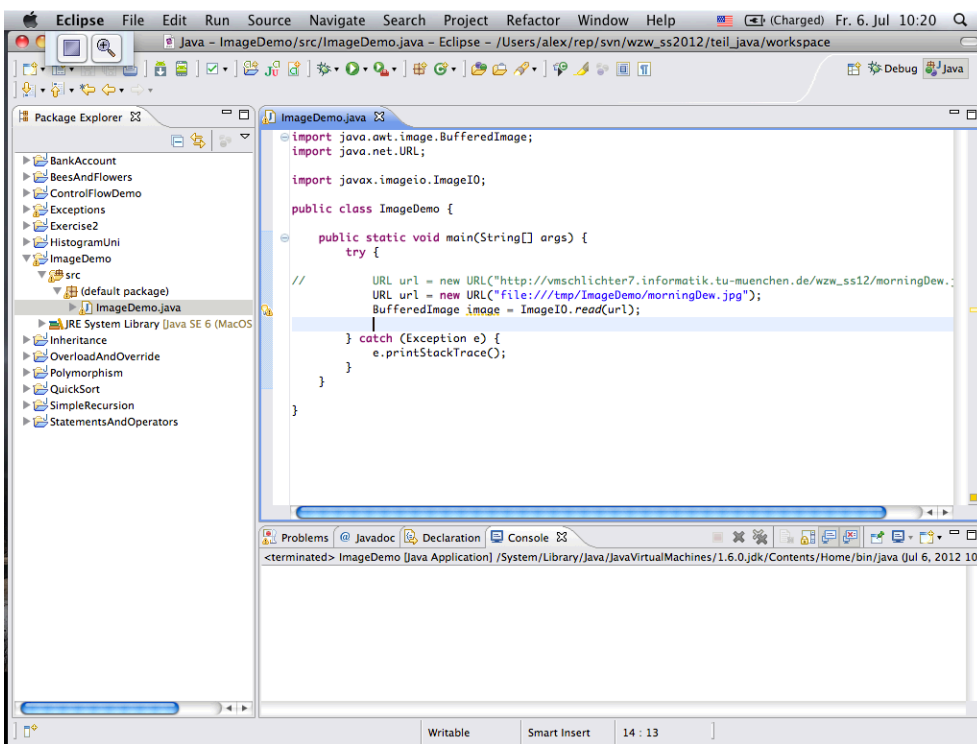
Problems @ Javadoc Declaration Console

```
<terminated> ExceptionsDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0_jdk/Contents/Home/bin/java [Jul 6, 2011]
Exception in thread "main" IllegalArgumentException
    at Bicycle.<init>(Bicycle.java:12)
    at ExceptionsDemo.main(ExceptionsDemo.java:24)
```

Writable Smart Insert 5 : 9







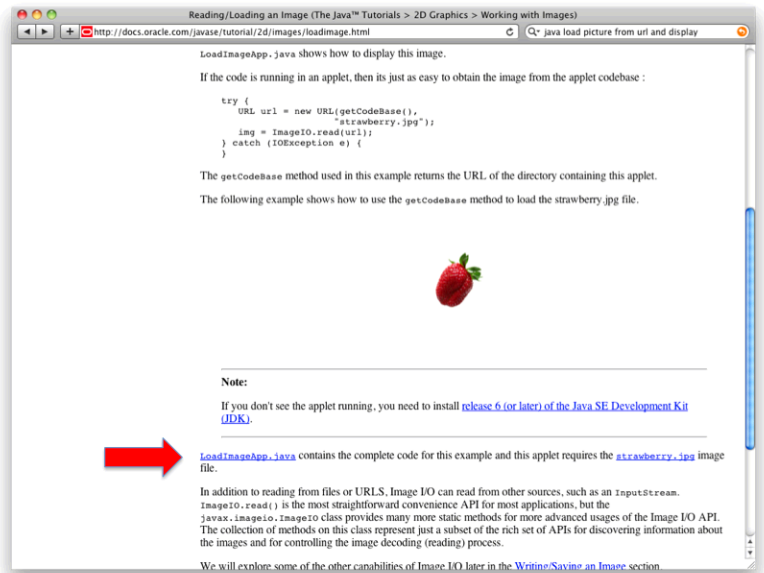
8 Solving a Problem

Example Task

So what about displaying the image we've just loaded?

8 Solving a Problem

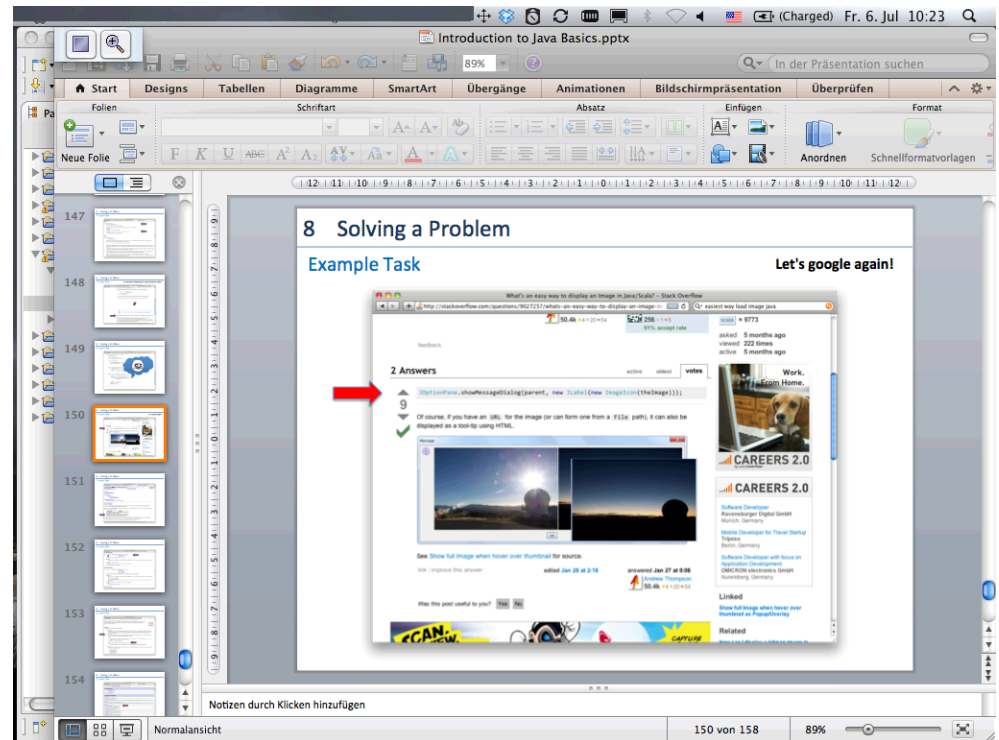
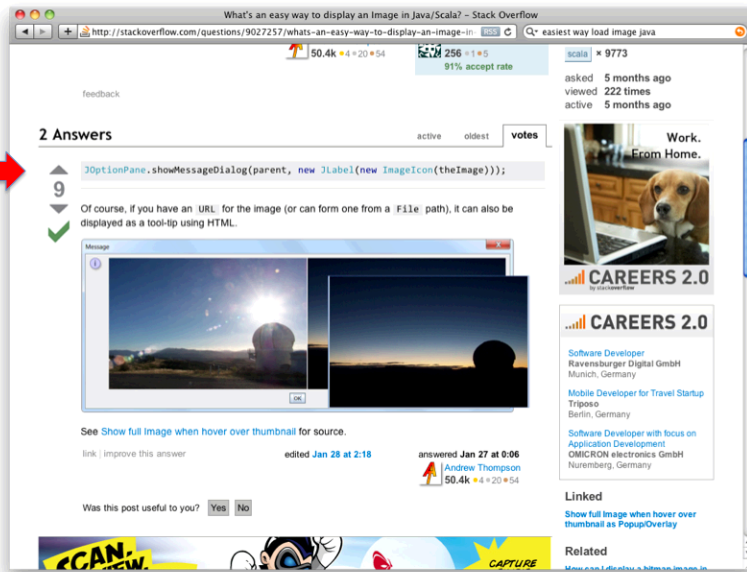
Example Task



8 Solving a Problem

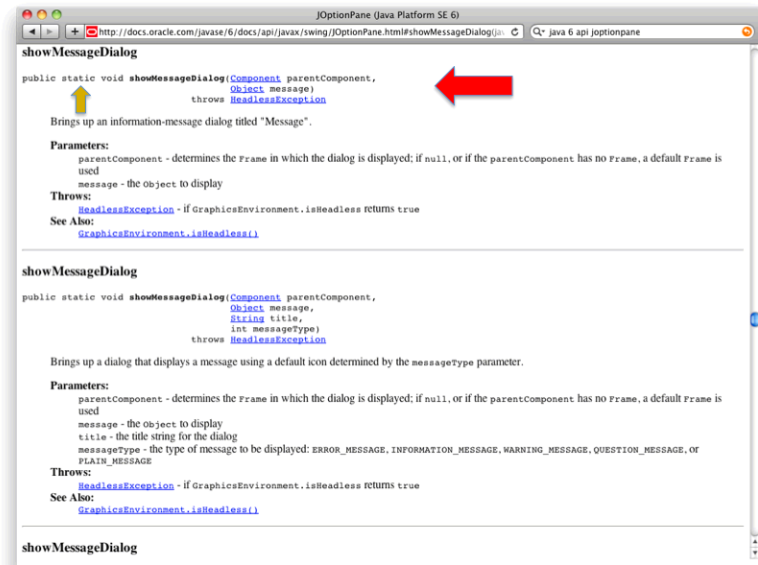
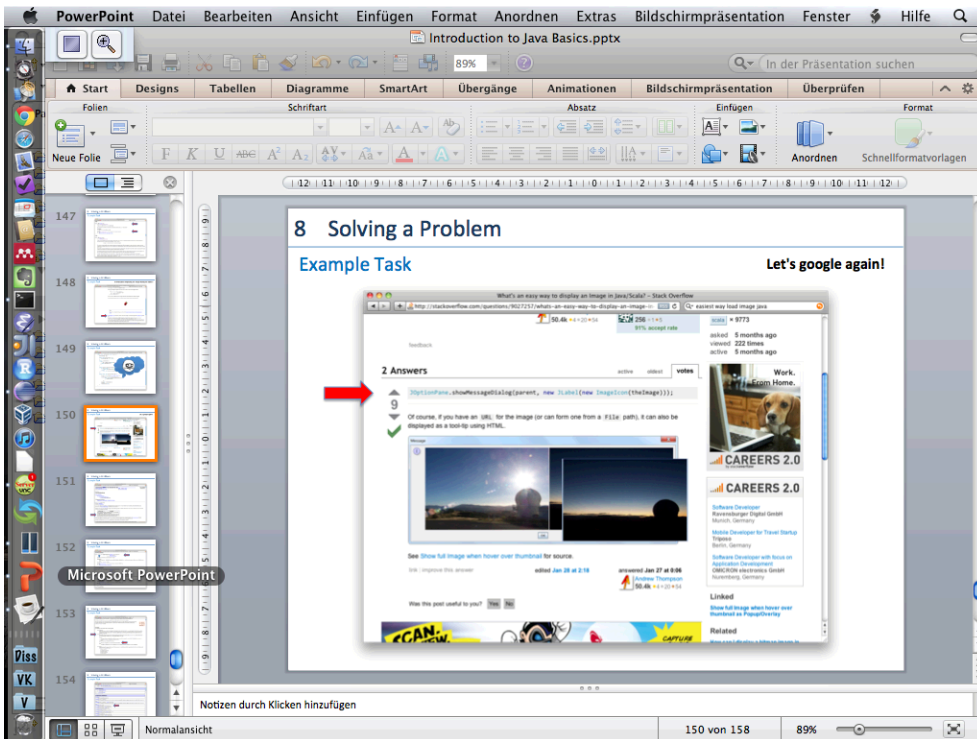
Example Task

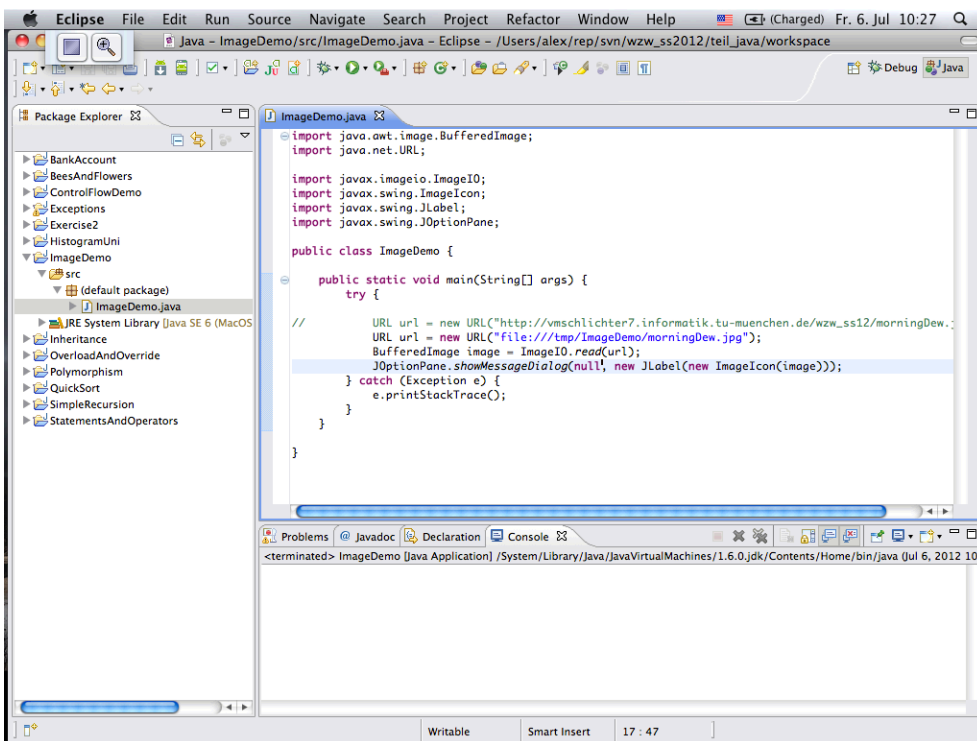
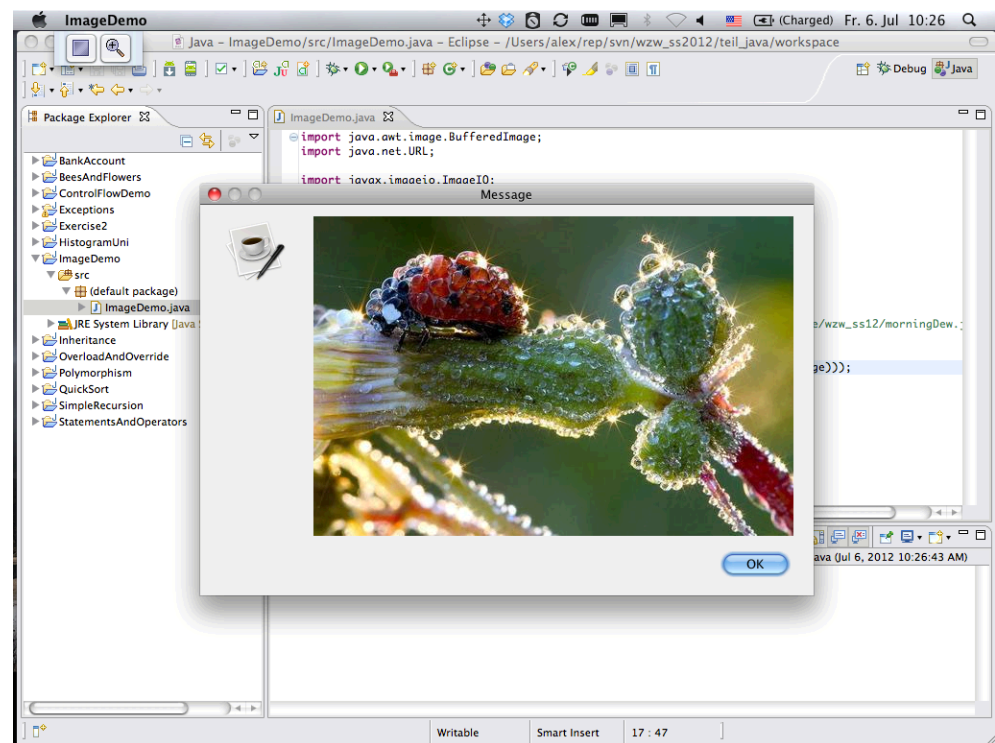
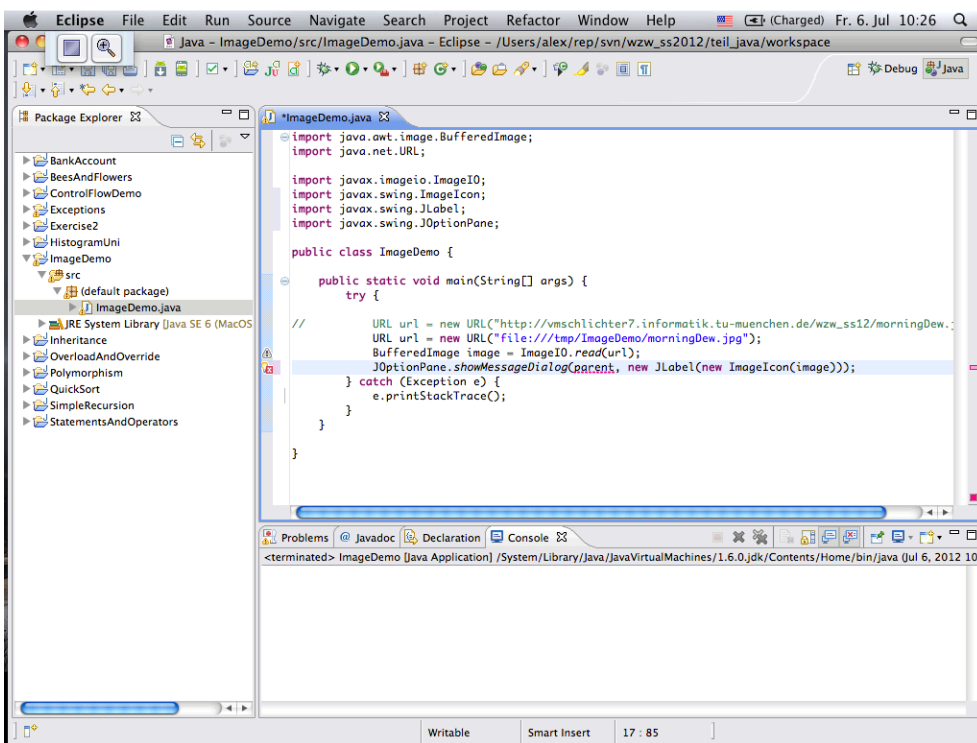
Let's google again!



8 Solving a Problem

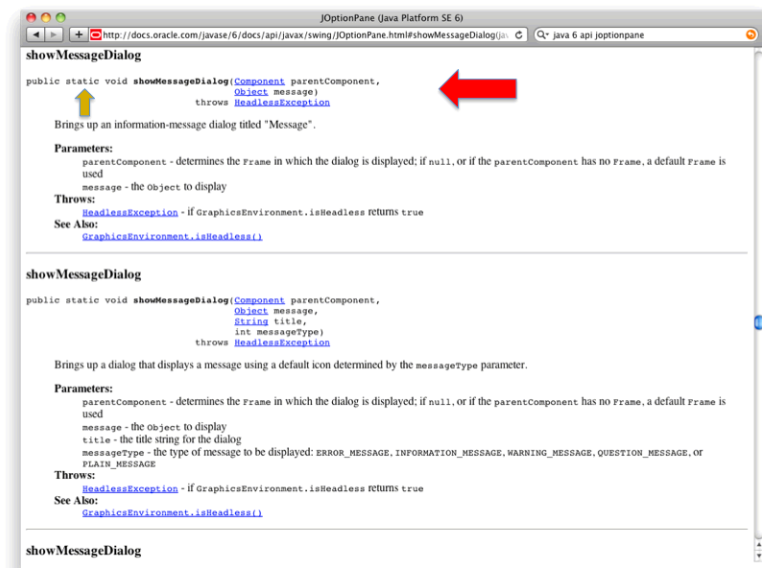
Example Task





8 Solving a Problem

Example Task



8 Solving a Problem

Example Task

convenience methods have also been defined -- overloaded versions of the basic methods that use different parameter lists.

All dialogs are modal. Each `show***Dialog` method blocks the caller until the user's interaction is complete.

The basic appearance of one of these dialog boxes is generally similar to the picture at the right, although the various look-and-feels are ultimately responsible for the final result. In particular, the look-and-feels will adjust the layout to accommodate the option pane's `componentOrientation` property.

icon	message
input value	
option buttons	

Parameters:
The parameters to these methods follow consistent patterns:

parentComponent
Defines the component that is to be the parent of this dialog box. It is used in two ways: the frame that contains it is used as the frame parent for the dialog box, and its screen coordinates are used in the placement of the dialog box. In general, the dialog box is placed just below the component. This parameter may be `null`, in which case a default frame is used as the parent, and the dialog will be centered on the screen (depending on the L&F).

message
A descriptive message to be placed in the dialog box. In the most common usage, message is just a `String` or `String` constant. However, the type of this parameter is actually `Object`. Its interpretation depends on its type:

Object[]
An array of objects is interpreted as a series of messages (one per object) arranged in a vertical stack. The interpretation is recursive -- each object in the array is interpreted according to its type.

Component
The component is displayed in the dialog.

Icon
The icon is wrapped in a `JLabel` and displayed in the dialog.

others
The object is converted to a `String` by calling its `toString` method. The result is wrapped in a `JLabel` and displayed.

messageType
Defines the style of the message. The Look and Feel manager may lay out the dialog differently depending on this value, and will often provide a default icon. The possible values are:

- `ERROR_MESSAGE`
- `INFORMATION_MESSAGE`
- `WARNING_MESSAGE`
- `QUESTION_MESSAGE`
- `PLAIN_MESSAGE`

8 Solving a Problem

Example Task

This class implements accessibility support for the `ImageIcon` class.

Field Summary

protected	component
protected	component
protected	tracker

Constructor Summary

`ImageIcon()`
Creates an uninitialized image icon.

`ImageIcon(byte[] imageData)`
Creates an `ImageIcon` from an array of bytes which were read from an image file containing a supported image format, such as GIF, JPEG, or (as of 1.3) PNG.

`ImageIcon(byte[] imageData, String description)`
Creates an `ImageIcon` from an array of bytes which were read from an image file containing a supported image format, such as GIF, JPEG, or (as of 1.3) PNG.

`ImageIcon(Image image)`
Creates an `ImageIcon` from an image object.

`ImageIcon(Image image, String description)`
Creates an `ImageIcon` from the image.

`ImageIcon(String filename)`
Creates an `ImageIcon` from the specified file.

`ImageIcon(String filename, String description)`
Creates an `ImageIcon` from the specified file.

`ImageIcon(URL location)`
Creates an `ImageIcon` from the specified URL.

`ImageIcon(URL location, String description)`
Creates an `ImageIcon` from the specified URL.

Method Summary

```
import java.awt.image.BufferedImage;
import java.net.URL;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

public class ImageDemo {

    public static void main(String[] args) {
        try {
            URL url = new URL("http://vmschlichter7.informatik.tu-muenchen.de/wzw_ss12/morningDew...");
            URL url = new URL("file:///tmp/ImageDemo/morningDew.jpg");
            BufferedImage image = ImageIO.read(url);
            JOptionPane.showMessageDialog(null, new JLabel(new ImageIcon(image)));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
import java.awt.image.BufferedImage;
import java.net.URL;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

public class ImageDemo {

    public static void main(String[] args) {
        try {
            URL url = new URL("http://vmschlichter7.informatik.tu-muenchen.de/wzw_ss12/morningDew...");
            URL url = new URL("file:///tmp/ImageDemo/morningDew.jpg");
            BufferedImage image = ImageIO.read(url);
            JOptionPane.showMessageDialog(null, new JLabel(new ImageIcon(image)));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

8 Solving a Problem

Example Task

Overview Package **Class** Use Tree Deprecated Index Help

java.awt.image

Class BufferedImage

java.lang.Object
↳ java.awt.Image
↳ java.awt.image.BufferedImage

All Implemented Interfaces:
RenderedImage, WritableRenderedImage, Transparency

public class BufferedImage
extends Image
implements WritableRenderedImage, Transparency

The BufferedImage subclass describes an **Image** with an accessible buffer of image data. A **BufferedImage** is comprised of a **ColorModel** and a **Raster** of image data. The number and types of bands in the **Raster** must match the number and types required by the **ColorModel** to represent its color and alpha components. All **BufferedImage** objects have an upper left corner coordinate of (0, 0). Any **Raster** used to construct a **BufferedImage** must therefore have **minX=0** and **minY=0**.

This class relies on the data fetching and setting methods of **Raster**, and on the color characterization methods of **ColorModel**.

See Also:
[ColorModel](#), [Raster](#), [WritableRaster](#)

Field Summary	
static int	TYPE_3BYTE_BGR Represents an image with 8-bit RGB color components, corresponding to a Windows-style BGR color model) with the colors Blue, Green, and Red stored in 3 bytes.
static int	TYPE_4BYTE_ABGR Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha.
static int	TYPE_4BYTE_ABGR_PSR Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha.

8 Solving a Problem

Example Task

```
import java.awt.image.BufferedImage;  
import java.io.IOException;  
import java.net.URL;  
  
import javax.swing.ImageIcon;  
import javax.swing.JLabel;  
import javax.swing.JOptionPane;  
import javax.swing.UIManager;  
  
public class ImageDemo {  
  
    public static void main(String[] args) {  
        try {  
  
            URL url = new URL("http://www.google.com/intl/en_ALL/images/logos/images_lg.gif");  
            BufferedImage image = ImageIO.read(url);  
            JOptionPane.showMessageDialog(null, new JLabel(new ImageIcon(image)));  
  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

image data. The number and types of bands in the **Raster** must match the number and types required by the **ColorModel** to represent its color and alpha components. All **BufferedImage** objects have an upper left corner coordinate of (0, 0). Any **Raster** used to construct a **BufferedImage** must therefore have **minX=0** and **minY=0**.

This class relies on the data fetching and setting methods of **Raster**, and on the color characterization methods of **ColorModel**.

See Also:
[ColorModel](#), [Raster](#), [WritableRaster](#)

Field Summary	
static int	TYPE_3BYTE_BGR Represents an image with 8-bit RGB color components, corresponding to a Windows-style BGR color model) with the colors Blue, Green, and Red stored in 3 bytes.
static int	TYPE_4BYTE_ABGR Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha.
static int	TYPE_4BYTE_ABGR_PSR Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha.

8 Solving a Problem

Example Task

Field Summary	
static int	TYPE_3BYTE_BGR Represents an image with 8-bit RGB color components, corresponding to a Windows-style BGR color model) with the colors Blue, Green, and Red stored in 3 bytes.
static int	TYPE_4BYTE_ABGR Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha.
static int	TYPE_4BYTE_ABGR_PSR Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha.

Constructor Summary	
ImageDemo()	Creates an uninitialized image icon.
ImageDemo(URL url)	Creates an ImageIcon from an array of bytes which were read from an image file containing a supported image format, such as GIF, JPEG, or (as of 1.3), PNG.
ImageDemo(URL url, BufferedImage description)	Creates an ImageIcon from an array of bytes which were read from an image file containing a supported image format, such as GIF, JPEG, or (as of 1.3), PNG.
ImageDemo(Image image)	Creates an ImageIcon from an image object.
ImageDemo(Image image, BufferedImage description)	Creates an ImageIcon from the image.
ImageDemo(String filename)	Creates an ImageIcon from the specified file.
ImageDemo(String filename, BufferedImage description)	Creates an ImageIcon from the specified file.
ImageDemo(String url)	Creates an ImageIcon from the specified URL.
ImageDemo(String url, BufferedImage description)	Creates an ImageIcon from the specified URL.

8 Solving a Problem

Example Task

Field Summary	
static int	TYPE_3BYTE_BGR Represents an image with 8-bit RGB color components, corresponding to a Windows-style BGR color model) with the colors Blue, Green, and Red stored in 3 bytes.
static int	TYPE_4BYTE_ABGR Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha.
static int	TYPE_4BYTE_ABGR_PSR Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha.

Constructor Summary	
ImageDemo()	Creates an uninitialized image icon.
ImageDemo(URL url)	Creates an ImageIcon from an array of bytes which were read from an image file containing a supported image format, such as GIF, JPEG, or (as of 1.3), PNG.
ImageDemo(URL url, BufferedImage description)	Creates an ImageIcon from an array of bytes which were read from an image file containing a supported image format, such as GIF, JPEG, or (as of 1.3), PNG.
ImageDemo(Image image)	Creates an ImageIcon from an image object.
ImageDemo(Image image, BufferedImage description)	Creates an ImageIcon from the image.
ImageDemo(String filename)	Creates an ImageIcon from the specified file.
ImageDemo(String filename, BufferedImage description)	Creates an ImageIcon from the specified file.
ImageDemo(String url)	Creates an ImageIcon from the specified URL.
ImageDemo(String url, BufferedImage description)	Creates an ImageIcon from the specified URL.

PowerPoint Datei Bearbeiten Ansicht Einfügen Format Anordnen Extras Bildschirmpräsentation Fenster Hilfe

Introduction to Java Basics.pptx In der Präsentation suchen

Start Designs Tabellen Diagramme SmartArt Übergänge Animationen Bildschirmpräsentation Überprüfen

Folien

Schriftart Absatz Einfügen Format

Neue Folie

152
153
154
155
156
157
158

References

- **[JTutorial]**
Java Tutorial <http://java.sun.com/docs/books/tutorial> (June-July 2012)
- **Code examples** taken from
 - [JTutorial], licensed under Sun's Code Sample License available at http://developers.sun.com/license/berkeley_license.html
 - Stack Overflow, <http://stackoverflow.com>

Notizen durch Klicken hinzufügen

Normalansicht 158 von 158 89%