Java Reloaded

JDK 7

by Deniz Oğuz

Objective

Learn what is included in JDK 7 and remember to use Google when required

JDK Release History

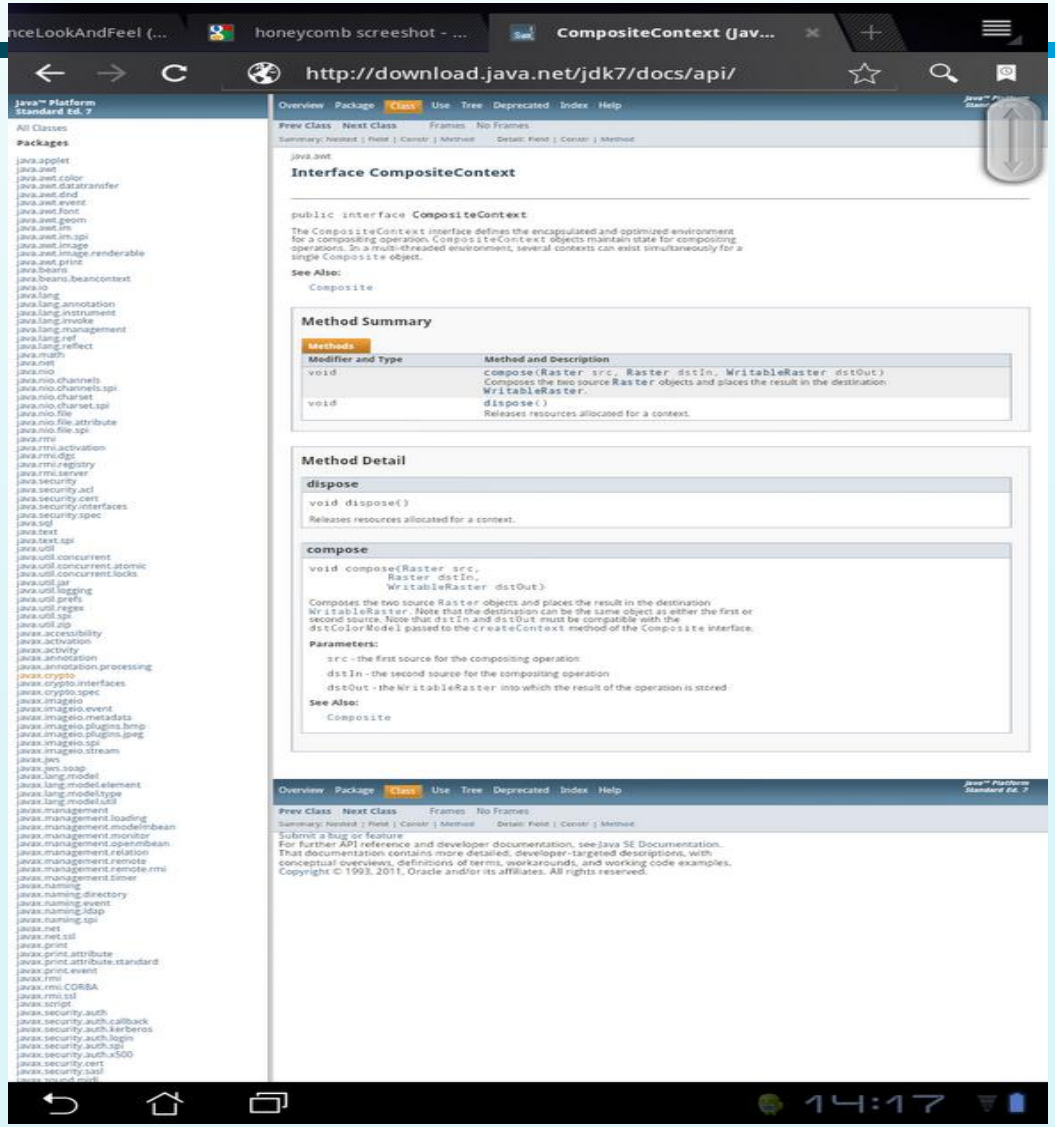
Version	Release Date
JDK 1.0	1996-01-23
JDK 1.1	1997-02-18
JDK 1.1.4	1997-09-12
JDK 1.1.5	1997-12-03
JDK 1.1.6	1998-04-24
JDK 1.1.7	1998-09-28
JDK 1.1.8	1999-04-08
J2SE 1.2	1998-12-04
J2SE 1.2.1	1999-03-30
J2SE 1.2.2	1999-07-08
J2SE 1.3	2000-05-08
J2SE 1.3.1	2001-05-17
J2SE 1.4.0	2002-02-13
J2SE 1.4.1	2002-09-16
J2SE 1.4.2	2003-06-26
J2SE 5.0	2004-09-29
J2SE 6	2006-12-11
Future Releases	
J2SE 7	2011-07-28
J2SE 8	Expected in late 2012

- Normal release cycle was 2 years, J2SE is 3 years late. It should have bean released at the end of 2008.
- Some of the most wanted features are postponed to J2SE 8.

How to Try JDK 7 Features?

1. Download JDK 7 build from jdk7.java.net
   - (download zipped version and extract to a folder)
2. Download Netbeans 7
   - (download SE version, 80 MB)
3. In the IDE, choose Tools > Java Platforms from the main menu
4. Click Add Platform and specify the directory that contains the JDK
5. Ensure JDK 1.7 is chosen in the Platforms list and click Close
6. On Project Properties' Libraries section Select JDK 1.7 as Java Platform
7. On Project Properties' Sources section Select JDK 7 as Source/Binary Format

New Javadoc Format



Project Coin

- String in switch statement
- Binary literals
- Underscore in literals
- Diamond operator
- Improved exception handling
- Try with resource
- Simplified vararg methods invocation

String in Switch Statement

- Before JDK 7 only byte, short, char, int were allowed in switch statements
- JDK 7 allows Strings to be used in switch statements
- Why isn't long supported in switch statements?

```
Console console = System.console();
String day= console.readLine();
switch (day) {
    case "monday" : console.writer().write("1");break;
    case "tuesday" : console.writer().write("2");break;
    case "wednesday" : console.writer().write("3");break;
    case "thursday" : console.writer().write("4");break;
    case "friday" : console.writer().write("5");break;
    case "saturday" : console.writer().write("6");break;
    case "sunday" : console.writer().write("7");break;
    default:console.writer().write("?");
}
console.flush();
```

Binary Literals

- It is now easier to specify numbers in binary form
  - int mask = 0b00000000000000000000000011111111;
  - int mask = Integer.parseInt("00000000000000000000000011111111", 2);
  - int mask = 255;

Underscores in Numbers

- int money = 100_000_000;
- long creditCNumber = 3434_3423_4343_4232L;
- int mask = 0b0000_0000_0000_0000_0000_0000_1111_1111;

Diamond Operator

- Before JDK 7:

```
Map<Integer, Track> trackStore = new ConcurrentHashMap<Integer, Track>();
```

- With JDK 7:

```
Map<Integer, Track> trackStore = new ConcurrentHashMap<>();
```

Multi Catch and Final Rethrow

Problem : You want to handle multiple exceptions with the same code block.

Before JDK 7

```
try {
    InputStream inStream = readStream(settingFile);
    Setting setting = parseFile(inStream);
} catch (IOException ex) {
    log.warn("Can not access file", settingFile);
} catch (FileNotFoundException ex) {
    log.warn("Can not access file", settingFile);
} catch (ParseException ex) {
    log.warn("{} has incorrect format:{}", settingFile, ex.getMessage());
}
```

With JDK 7

```
try {
    InputStream inStream = readStream(settingFile);
    Setting setting = parseFile(inStream);
} catch (IOException | FileNotFoundException ex) {
    log.warn("Can not access file", settingFile);
} catch (ParseException ex) {
    log.warn("{} has incorrect format:{}", settingFile, ex.getMessage());
}
```

Multi Catch and Final Re-throw Cont.

Problem : You want to handle multiple exceptions with the same code block.

```
public  Setting readSettings(Strin settingFile) throws ParseException, IOException,
FileNotFoundException {
    try {
        InputStream inStream = readStream(settingFile);
        Setting setting = parseFile(inStream);
    } catch (Throwable ex) {
        log.warn("Can not read settings", settingFile);
        throw ex;
    }
    ……..
}
```

Try With Resource

```
private static String readConfiguration(String file) {
    BufferedReader reader = null;
    try {
        reader = new BufferedReader(new FileReader(file));
        String line = null;
        StringBuilder content = new StringBuilder(1000);
        while ((line = reader.readLine()) != null) {
            content.append(line);
        }
        return content.toString();
    } catch (IOException ex){
        throw new ConfigurationException("Can not read configuration file:{}", file);
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

Before JDK 7

Try With Resource Cont'

New Try Syntax

```
private static String readConfigurationNew(String file) {
    try (BufferedReader reader = new BufferedReader(new FileReader(file));) {
        String line = null;
        StringBuilder content = new StringBuilder(1000);
        while ((line = reader.readLine()) != null) {
            content.append(line);
        }
        return content.toString();
    } catch (IOException ex){
        throw new ConfigurationException("Can not read configuration file:{}", file);
    }
}
```

With JDK 7

Finally Block Gone

Try With Resource and Autocloseable

- A new interface AutoCloseable is introduced
- Existing Closeable interface is changed to extend AutoCloseable interface
- A new method addSuppressed(Exception) is added to Throwable
- Exceptions throwed from close method of AutoCloseable are suppressed in favor of exceptions throwed from try-catch block
- See JavaDoc of Autocloseable for more detail

Autocloseable and Suppressed Exception

Exceptions from try-catch body suppresses exceptions thrown from AutoCloseable.close

```
public ABCResource implements AutoCloseable {
  @Override
   public void close() throws ABCException {
      dosomething();
      if (errorCondition) {
         throw new ABCException("Not supported yet.");
      }
   }
}
```

```
public ClientClass {
   public void useABCResource() {
      try (ABCResource resource = new ABCResource();) {
         dosomething();
         if (errorCondition) {
           throw new ClientException("Not supported yet.");
         }
      }
   }
}
```

What happens if Autocloseable throws an exception after the ClientException in try-catch block?

Better Support for Other Languages in JVM

- Statically Typed vs Dynamically Typed
  - Compile time type checking vs runtime type checking
  - Java, C, Scala are examples of statically typed languages
  - Java Script, Ruby are examples of dynamically typed languages
- Strongly Typed vs Weakly Typed
  - Automatic type conversion as necessary vs fixed type
  - Java Script is weakly typed language
  - Java is a strongly typed language

```
public void printTotal(a, b) {
   print a + b;
}
```

What does + means is decided at runtime during execution.

```
public void printTotal(int a, int b) {
   print a + b;
}
```

What does + means is decided at compile time.

Dynamic Typing was Difficult to Implement on JVM

- New bytecode is introduced
  - invokeinterface
  - invokestatic
  - invokevirtual
  - invokespecial
  -  invokedynamic (Only bytecode in JVM that is not used by Java PL)

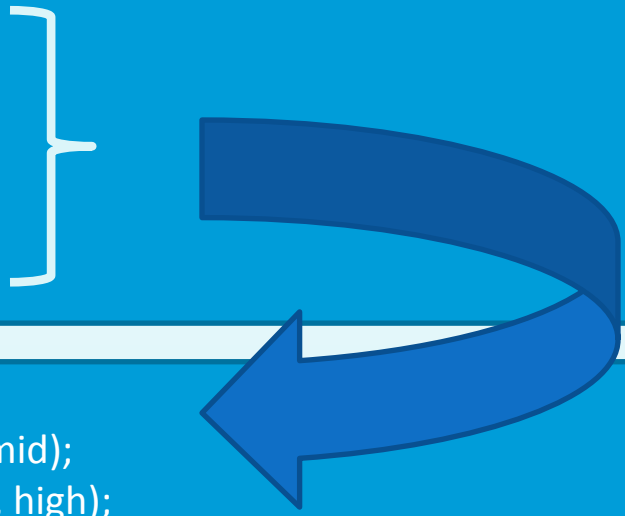
- Execution environment of the programming language provides a bootstrap method for resolving method invocations

Fork/Join Framework

- Uses ***work-stealing*** algorithm
- Task is broken into smaller parts recursively
- A new ExecutorService implementation ForkJoinPool is added
- ForkJoinTask and its subclasses RecursiveAction and RecursiveTask are added

General Usage Pattern of Fork/Join

```
Result compute(Problem problem) {
  if (problem is small)
    directly solve problem
  else {
    split problem into independent parts
    fork new subtasks to solve each part
    join all subtasks
    compose result from subresults
  }
```

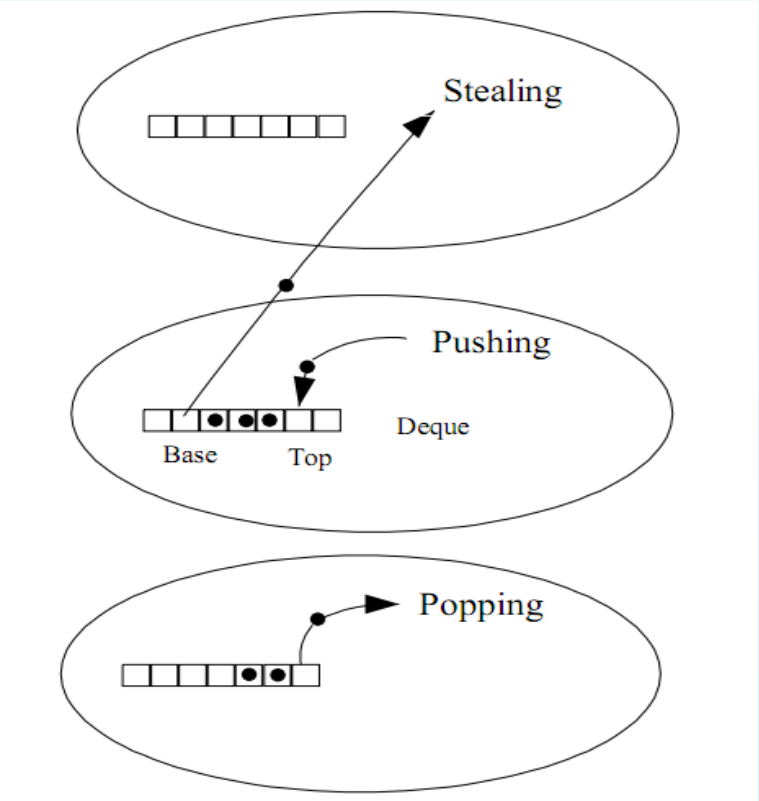


```
else {
    RecursiveTask left = new ComputationTask(array, low, mid);
    RecursiveTask right = new ComputationTask(array, mid, high);
    right.fork();
    return left.compute() + right.join();
}
```

```
main {
    RecursiveTask computationTask = new ComputationTask(array, 0, array.length);
    ForkJoinPool mainPool = new ForkJoinPool();
    Long result = mainPool.invoke(computationTask);
}
```

Work Stealing

- Less thread contention
- Improved data locality
- Execute large tasks early



TransferQueue

Allows producers to wait until message is processed by a consumer even if the queue is not full.

- **transfer(E e)**
  - Transfers the element to a consumer, waiting if necessary to do so.
- **tryTransfer(E e)**
  - Transfers the element to a waiting consumer immediately, if possible.
- **tryTransfer**(**E** e, long timeout, TimeUnit unit)
  - Transfers the element to a consumer if it is possible to do so before the timeout elapses.
- **getWaitingConsumerCount**()
  - Returns an estimate of the number of consumers waiting to receive elements via BlockingQueue.take() or timed poll
- **hasWaitingConsumer**()
  - Returns true if there is at least one consumer waiting to receive an element via BlockingQueue.take () or timed poll.

ThreadLocalRandom

- A random number generator isolated to current thread
- Aim is to reduce contention in multi threaded environments
- Usage:
  - ThreadLocalRandom.current().nextInt(min, max);
  - ThreadLocalRandom.current().nextLong(min, max);
  - ThreadLocalRandom.current().nextDouble(min, max);

ConcurrentLinkedDeque

- Unbound concurrent deque based on linked nodes
- Concurrent insertion, removal, and access operations execute safely across multiple threads
- Iterators are weakly consistent and do not throw *ConcurrentModificationException*
- size() method is NOT constant in time
- Bulk operations are not guaranteed to perform atomically

Phaser

- An reusable synchronization barrier
- Similar to CyclicBarrier or CountDownLatch but with more advance features :
  - Allows number of registered parties to change after Phaser creation
  - Each generation increment phase number of phaser
  - There are blocking and non blocking versions of operations
- Extremely flexible and complex, use Javadoc when you need to use this class

SCTP Support

Feature	SCTP	TCP	UDP
Connection-oriented	✓	✓	
Full duplex	✓	✓	✓
Reliable data transfer	✓	✓	
Partial-reliable data transfer	optional		
Ordered data delivery	✓	✓	
Unordered data delivery	✓		✓
Flow control	✓	✓	
Congestion control	✓	✓	
ECN capable	✓	✓	
Selective ACKs	✓	optional	
Preservation of message boundaries	✓		✓
Path MTU discovery	✓	✓	
Application PDU fragmentation	✓	✓	
Application PDU bundling	✓	✓	
Multistreaming	✓		
Multihoming	✓		
Protection against SYN flooding attacks	✓		n/a
Allows half-closed connections		✓	n/a
Reachability check	✓	✓	
Psuedo-header for checksum	(uses vtags)	✓	✓
Time wait state	for vtags	for 4-tuple	n/a

- Not all operating systems support SCTP

The Need for Java NIO.2

- Methods works inconsistently
  - Delete method sometimes can not delete
  - Rename method sometimes can not rename
- No Exception is thrown from failed method
- Accessing metadata of files is limited
- Does not scale well
  - Listing a directory may take a long time (especially over network directories)
- A change notification facility is not provided
- Developers wanted to create their own file system implementations
  - For example an in-memory file system

Pluggable FileSystems

- java.nio.file.FileSystems is factory for file systems.
- Implement java.nio.file.spi.FileSystemProvider to provide your own file systems
  - A filesystem provider for Zip and Jar files are included in JDK 7
  - You can implement a filesystem to open a ISO image as a file system, or implement a RAM disk etc……
- java.nio.file.FileSystems.getDefault() is used most of the time
- Multiple/Alternate views of same underlying files
  - Hides some files for security, read-only view, etc.

java.nio.file.Path and java.nio.file.Files

- Use java.nio.file.Path and java.nio.file.Files instead of java.io.File
  - Use java.io.File.toPath and java.nio.file.Path.toFile methods to integrate with legacy code
- java.nio.file.Paths contains factory methods for java.nio.file.Path
  - static Path getPath(String first, String… more)
  - static Path getPath(URI uri)
- Once you obtained Path object use java.nio.file.Files static methods to process
  - copy, createLink, createTempFile, delete, exist, getPosixFilePermissions, getAttribute, isHidden, isExecutable, isSymbolicLink, newByteChannel ……

Asynchronous I/O

- Allows application to continue on something while waiting for I/O
- Asynchronous I/O operations will usually take one of two forms:
  - Future<V> *operation*(...)
  - void *operation*(... A attachment, CompletionHandler<V,? super A> handler)
- See classes AsynchronousXXXXChannel that extends AsynchronousChannel

```
Path path = Paths.get("/home/deniz/dropbox/Getting Started.pdf");
AsynchronousFileChannel ch = AsynchronousFileChannel.open(path);
ByteBuffer buf = ByteBuffer.allocate(1024);
Future<Integer> result = ch.read(buf, 0);  //read does not block
while (!result.isDone()) {
   System.out.println("lets do something else while waiting");
}
System.out.println("Bytes read = " + result.get()); //Future.get will block
ch.close();
```

Watch Service

- WatchService allows you to monitor a watchable object for changes and events. It implements Reactor pattern. Just like Selector does for Channels.
- Use Watchable.register to register with java.nio.file.WatchService to listen changes

  WatchKey register(WatchService watcher, WatchEvent.Kind<?>... events)

  - WatchEvent.Kind is an interface, actual applicable alternatives depends on Watchable
- Example Watchable objects
  - Path, FileSystem

```
Path path = Paths.get("C:/Users/Deniz/Dropbox/");
WatchService watcher = path.getFileSystem().newWatchService();
while (true) {
   path.register(watcher, StandardWatchEventKinds.ENTRY_CREATE);
   WatchKey key = watcher.take();  // block for event
   for (WatchEvent event : key.pollEvents()) {
      Path pathOfNewFile = (Path) event.context(); // path of new file
      System.out.println("File is created:" + pathOfNewFile.toString());
   }
   key.cancel();
}
```

FileVisitor

```
Path start = ...
Files.walkFileTree(start, new SimpleFileVisitor<Path>() {
  public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
      Files.delete(file);
      return FileVisitResult.CONTINUE;
   }
   public FileVisitResult postVisitDirectory(Path dir, IOException e) throws IOException {
      if (e == null) {
         Files.delete(dir);
         return FileVisitResult.CONTINUE;
      } else {
         // directory iteration failed
        throw e;
      }
   }});
```

FileTypeDetector

```
Files.walkFileTree(Paths.get("C:/Users/Deniz/Downloads"), new SimpleFileVisitor<Path>() {
    public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
        String fileType = Files.probeContentType(file);
        System.out.println(file.toString() + ":" + fileType);
        return FileVisitResult.CONTINUE;
    }
 });
```

Output:

C:\Users\Deniz\Downloads\Wirofon-Client-0.2.10-rc01.exe:application/x-msdownload

C:\Users\Deniz\Downloads\Plan B-She Said.mp3:audio/mpeg

C:\Users\Deniz\Downloads\ROM or Kernel-Update.pdf:application/pdf

Xrender Pipeline

- Will replace OpenGL pipeline on Unix, Linux platforms
  - Due to poor OpenGL drivers OpenGL pipeline has several problems
  - A lot of drivers has optimized Xrender implementations
  - Xrender is more suited 2D applications than OpenGL
  - Xrender applications are native X11 applications
  - Other GUI libraries are also using Xrender for 2D effects
    - QT4, GTK+, KDE4
- Use -Dsun.java2d.xrender=true to enable it

Translucent Windows

- Java SE 6u10 introduced com.sun.awt.AWTUtilities to support translucent and shaped windows
  - AWTUtilities is removed in JDK7
- Simple translucency

```
setUndecorated(true);
setOpacity (opacityValue); //opacity value is between 0.0 – 1.0
```

- Per-Pixel translucency

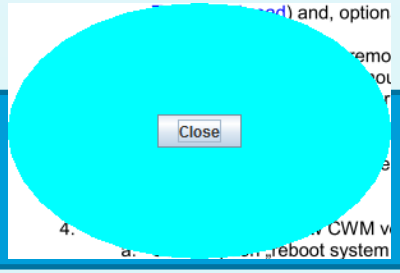
```
Color colorA = new Color (255, 0, 0, 0);     // Transparent red
Color colorB = new Color (255, 0, 0, 255); // Solid red
setBackground(new Color (0, 0, 0, 0)); // activate per-pixel
                                                    // translucency.

protected void paintComponent (Graphics g) {
   Graphics2D g2d = (Graphics2D) g;
   GradientPaint gp = new GradientPaint (
                0.0f, 0.0f, colorA, 0.0f, getHeight (), colorB, true);
   g2d.setPaint (gp);
   g2d.fillRect (0, 0, getWidth (), getHeight ());
}
```

Shaped Windows

- Shaped windows
  - Only the parts that belong to the given Shape remain visible and clickable
- A shaped window can also be translucent if you want

```
setUndecorated(true);
setShape(new Ellipse2D.Float (0, 0, getWidth (), getHeight ()));
```

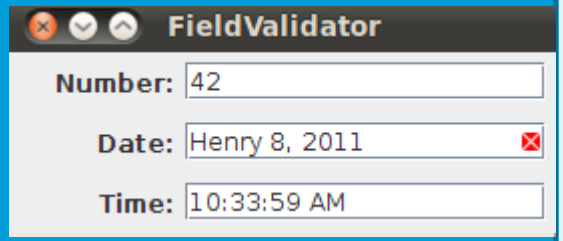


For code samples see: Exploring JDK 7, Part 2: Translucent and Shaped Windows

JLayer and LayerUI

- Similar to GlassPane but it is a layer around any JComponent not only JRootPane
- You can mask mouse events in installUI method

```
LayerUI<JFormattedTextField> layerUI = new ValidationLayerUI();
……
JFormattedTextField dateField = new JFormattedTextField(dateFormat);
JFormattedTextField numberField = new JFormattedTextField(numberFormat);
……
JPanel datePanel = new JPanel();
……
datePanel.add(new JLayer<JFormattedTextField>(dateField, layerUI));
datePanel.add(new JLayer<JFormattedTextField>(numberField , layerUI));
……
class ValidationLayerUI extends LayerUI<JFormattedTextField> {
   public void paint (Graphics g, JComponent c) {
       super.paint (g, c);
       JLayer jlayer = (JLayer) c;
       JFormattedTextField ftf = (JFormattedTextField)jlayer.getView();
       if (!ftf.isEditValid()) {
          //paint an error indicator using Graphics
       }
   }
}
//See http://download.oracle.com/javase/tutorial/uiswing/misc/jlayer.html for details
```



java.util.Objects

- Static utility methods for simplifying common operations on objects
- requireNonNull(T obj, String message)
  - Checks that the specified object reference is not null and throws a customized NullPointerException if it is
- int **compare**(T a, T b, Comparator<? super T> c)
  - Perform null and equals checks before invoking c.compare(a,b)
- boolean deepEquals(Object a, Object b)

  boolean equals(Object a, Object b)
  - Perform null checks and Arrays.deepEquals if necessary
- String toString(Object o, String nullDefault)
  - Perform null check and return nullDefault if o is null
- There are other similar methods, check javadoc

ThreadLocal

- These variables differ from their normal counterparts in that each thread that accesses one (via its get or set method) has its own, independently initialized copy of the variable

```
import java.util.concurrent.atomic.AtomicInteger;
public class UniqueThreadIdGenerator {
   private static final AtomicInteger uniqueId = new AtomicInteger(0);
   private static final ThreadLocal<Integer> uniqueNum = new ThreadLocal<Integer> () {
      @Override
      protected Integer initialValue() {
         return uniqueId.getAndIncrement();
      }
   };
  public static int getCurrentThreadId() {
      return uniqueId.get();
  }
} // UniqueThreadIdGenerator
```

Executed for each thread that calls uniqueId.get()

Locale Category

- Locale.setDefault(Category, Locale)
- Locale Locale.getDefault(Category)
- Category.DISPLAY
  - Category used to represent the default locale for displaying user interfaces
- Category.FORMAT
  - Category used to represent the default locale for formatting dates, numbers, and/or currencies
- What about default locale Locale.getDefault() ?
  - It stays as a different locale (not display or format)
  - But Locale.setDefault(Locale) sets 3 different locales now:display, format and default locales
  - Default locale depends on OS environment variables or value of

References

- Exploring JDK 7, Part 2: Translucent and Shaped Windows
- What's New in NIO.2 (pdf)
- JDK 7 Features
- The Java NIO.2 File System in JDK 7
- Xrender Proposal
- Java 2D Enhancements in Java SE 7
- How to Decorate Components with the JLayer Class
- StackOverflow