

---

# 13

In this chapter:

- *AWTException*
- *IllegalComponentStateException*
- *AWTError*

## *AWT Exceptions and Errors*

This chapter describes `AWTException`, `IllegalComponentStateException`, and `AWTError`. `AWTException` is a subclass of `Exception`. It is not used by any of the public classes in `java.awt`; you may, however, find it convenient to throw `AWTException` within your own code. `IllegalComponentStateException` is another `Exception` subclass, which is new to Java 1.1. This exception is used when you try to do something with a `Component` that is not yet appropriate. `AWTError` is a subclass of `Error` that is thrown when a serious problem occurs in AWT—for example, the environment is unable to get the platform's Toolkit.

### *13.1 AWTException*

`AWTException` is a generic exception that can be thrown when an exceptional condition has occurred within AWT. None of the AWT classes throw this. If you subclass any of the AWT classes, you can throw an `AWTException` to indicate a problem. Using `AWTException` is slightly preferable to creating your own `Exception` subclass because you do not have to generate another class file. Since it is a part of Java, `AWTException` is guaranteed to exist on the run-time platform.

If you throw an instance of `AWTException`, like any other `Exception`, it must be caught in a catch clause or declared in the throws clause of the method.

#### *13.1.1 AWTException Method*

##### *Constructor*

*public AWTException (String message)*

The sole constructor creates an `AWTException` with a detailed message of message. This message can be retrieved using `getMessage()`, which it inherits from `Exception` (and which is required by the `Throwable` interface). If you do

not want a detailed message, message may be null.

### 13.1.2 Throwing an *AWTException*

An *AWTException* is used the same way as any other *Throwable* object. Here's an example:

```
if (someProblem) {
    throw new AWTException ("Problem Encountered While Initializing");
}
```

## 13.2 *IllegalComponentStateException*

*IllegalComponentStateException* is a subclass of *IllegalStateException*; both are new to Java 1.1. This exception is used when you try to do something with a *Component* that is not yet appropriate. With the standard AWT components, this can happen only in three instances:

- If you call *setCaretPosition()* to set the cursor position of a text component before the component's peer exists.
- If you call *getLocale()* to get the locale of a component that does not have one and is not in a container that has one.
- If you call *getLocationOnScreen()* for a component that is not showing.

In these cases, the operation isn't fundamentally illegal; you are just trying to perform it before the component is ready. When you create your own components, you should consider using this exception for similar cases.

Since *IllegalComponentStateException* is a subclass of *RuntimeException*, you do not have to enclose method calls that might throw this exception within *try/catch* blocks. However, catching this exception isn't a bad idea, since it should be fairly easy to correct the problem and retry the operation.

### 13.2.1 *IllegalComponentStateException Method*

#### *Constructor*

*public IllegalComponentStateException ()* ★

The first constructor creates an *IllegalComponentStateException* instance with no detail message.

*public IllegalComponentStateException (String message) ★*

This constructor creates an `IllegalComponentStateException` with a detail message of `message`. This message can be retrieved using `getMessage()`, which it inherits from `Exception` (and is required by the `Throwable` interface).

### 13.2.2 *IllegalComponentStateException Example*

The following code throws an `IllegalComponentStateException`. The `Exception` occurs because the `TextField` peer does not exist when `setCaretPosition()` is called. `setCaretPosition()` throws an `IllegalComponentStateException`, and the next statement never executes.

```
import java.awt.TextField;
public class illegal {
    public static void main (String[] args) {
        new TextField().setCaretPosition (24);
        System.out.println ("Never gets here");
    }
}
```

## 13.3 *AWTError*

`AWTError` is a subclass of `Error` that is used when a serious run-time error has occurred within AWT. For example, an `AWTError` is thrown if the default `Toolkit` cannot be initialized or if you try to create a `FileDialog` within Netscape Navigator (since that program does not permit local file system access). When an `AWTError` is thrown and not caught, the virtual machine stops your program. You may throw this `Error` to indicate a serious run-time problem in any subclass of the AWT classes. Using `AWTError` is slightly preferable to creating your own `Error` because you don't have to provide another class file. Since it is part of Java, `AWTError` is guaranteed to exist on the run-time platform.

Methods are not required to declare that they throw `AWTError`. If you throw an error that is not caught, it will eventually propagate to the top level of the system.

### 13.3.1 *AWTError Method*

#### *Constructor*

*public AWTError (String message)*

The sole constructor creates an `AWTError` with a detail message of `message`. This message can be retrieved using `getMessage()`, which it inherits from `Error` (and is required by the `Throwable` interface). If you do not want a detailed message, `message` may be `null`.

### 13.3.2 Throwing an *AWTError*

The code in Example 13-1 throws an `AWTError` if it is executed with this command:

```
java -Dawt.toolkit=foo throwme
```

The error occurs because the Java interpreter tries to use the toolkit `foo`, which does not exist (assuming that class `foo` does not exist in your `CLASSPATH`). Therefore, `getDefaultToolkit()` throws an `AWTError`, and the next statement never executes.

*Example 13-1: The `throwme` class*

```
import java.awt.Toolkit;
public class throwme {
    public static void main (String[] args) {
        System.out.println (Toolkit.getDefaultToolkit());
        System.out.println ("Never Gets Here");
    }
}
```