Java Overview

An introduction to the Java Programming Language

Produced by:

Eamonn de Leastar (edeleastar@wit.ie)

Dr. Siobhan Drohan (sdrohan@wit.ie)

Essential Java

- **Overview**
  - Introduction
  - Syntax
  - Basics
  - Arrays
- **Classes**
  - Classes Structure
  - Static Members
  - Commonly used Classes
- **Control Statements**
  - Control Statement Types
  - If, else, switch
  - For, while, do-while

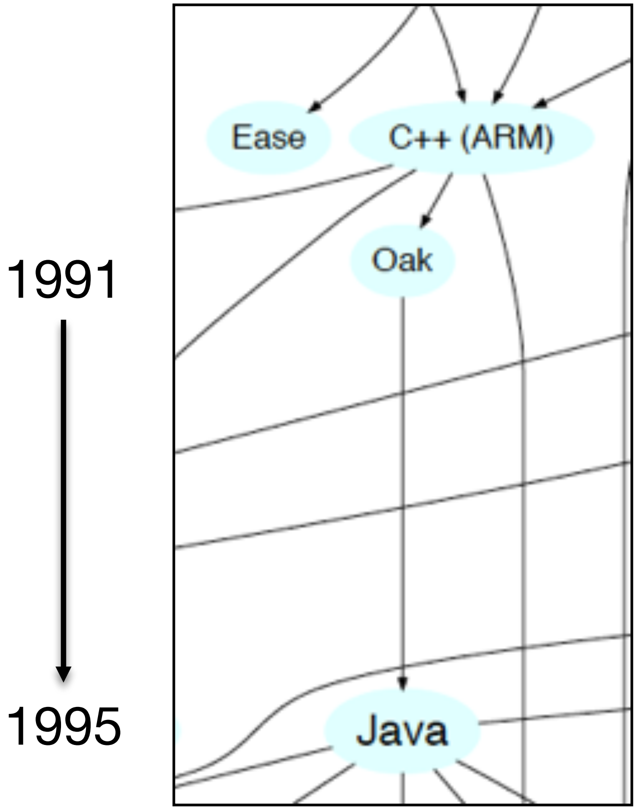
- **Inheritance**
  - Class hierarchies
  - Method lookup in Java
  - Use of this and super
  - Constructors and inheritance
  - Abstract classes and methods
  - Interfaces
- **Collections**
  - ArrayList
  - HashMap
  - Iterator
  - Vector
  - Enumeration
  - Hashtable

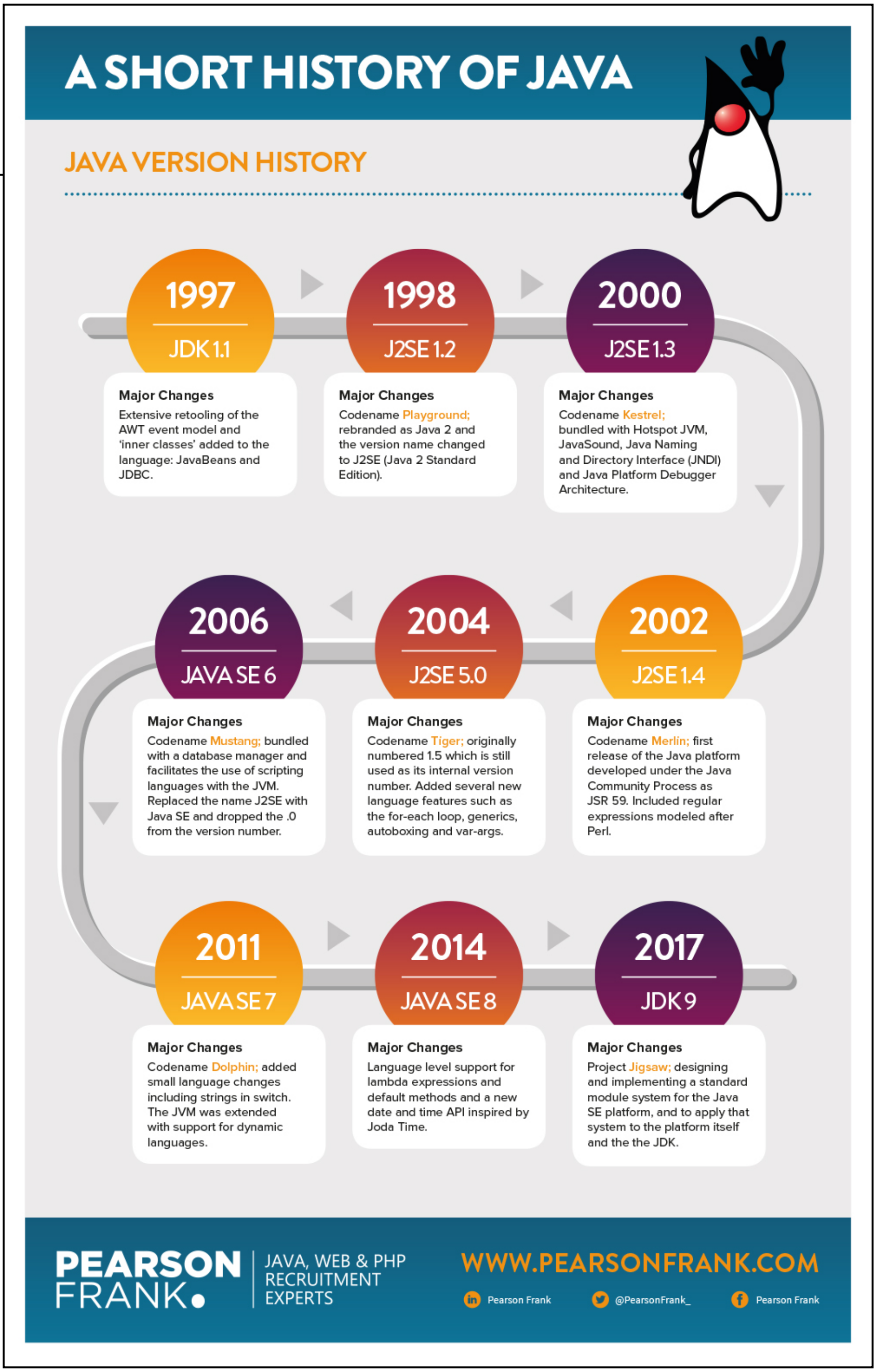
- **Exceptions**
  - Exception types
  - Exception Hierarchy
  - Catching exceptions
  - Throwing exceptions
  - Defining exceptions
  - Common exceptions and errors
- **Streams**
  - Stream types
  - Character streams
  - Byte streams
  - Filter streams
  - Object Serialization

Overview: Road Map

- Java Introduction
  - History
  - Portability
  - Compiler
  - Java Virtual Machine
  - Garbage collection
- Java Syntax
  - Identifiers
  - Expressions
  - Comments
- Java Basics
  - Java types
  - Primitives
  - Objects
  - Variables
  - Operators
  - Identity and equality
- Arrays
  - What are arrays?
  - Creating arrays
  - Using arrays

Java History





A SHORT HISTORY OF JAVA

JAVA VERSION HISTORY

1997 — JDK 1.1

Major Changes
Extensive retooling of the AWT event model and 'inner classes' added to the language: JavaBeans and JDBC.

1998 — J2SE 1.2

Major Changes
Codename Playground; rebranded as Java 2 and the version name changed to J2SE (Java 2 Standard Edition).

2000 — J2SE 1.3

Major Changes
Codename Kestrel; bundled with Hotspot JVM, JavaSound, Java Naming and Directory Interface (JNDI) and Java Platform Debugger Architecture.

2002 — J2SE 1.4

Major Changes
Codename Merlin; first release of the Java platform developed under the Java Community Process as JSR 59. Included regular expressions modeled after Perl.

2004 — J2SE 5.0

Major Changes
Codename Tiger; originally numbered 1.5 which is still used as its internal version number. Added several new language features such as the for-each loop, generics, autoboxing and var-args.

2006 — JAVA SE 6

Major Changes
Codename Mustang; bundled with a database manager and facilitates the use of scripting languages with the JVM. Replaced the name J2SE with Java SE and dropped the .0 from the version number.

2011 — JAVA SE 7

Major Changes
Codename Dolphin; added small language changes including strings in switch. The JVM was extended with support for dynamic languages.

2014 — JAVA SE 8

Major Changes
Language level support for lambda expressions and default methods and a new date and time API inspired by Joda Time.

2017 — JDK 9

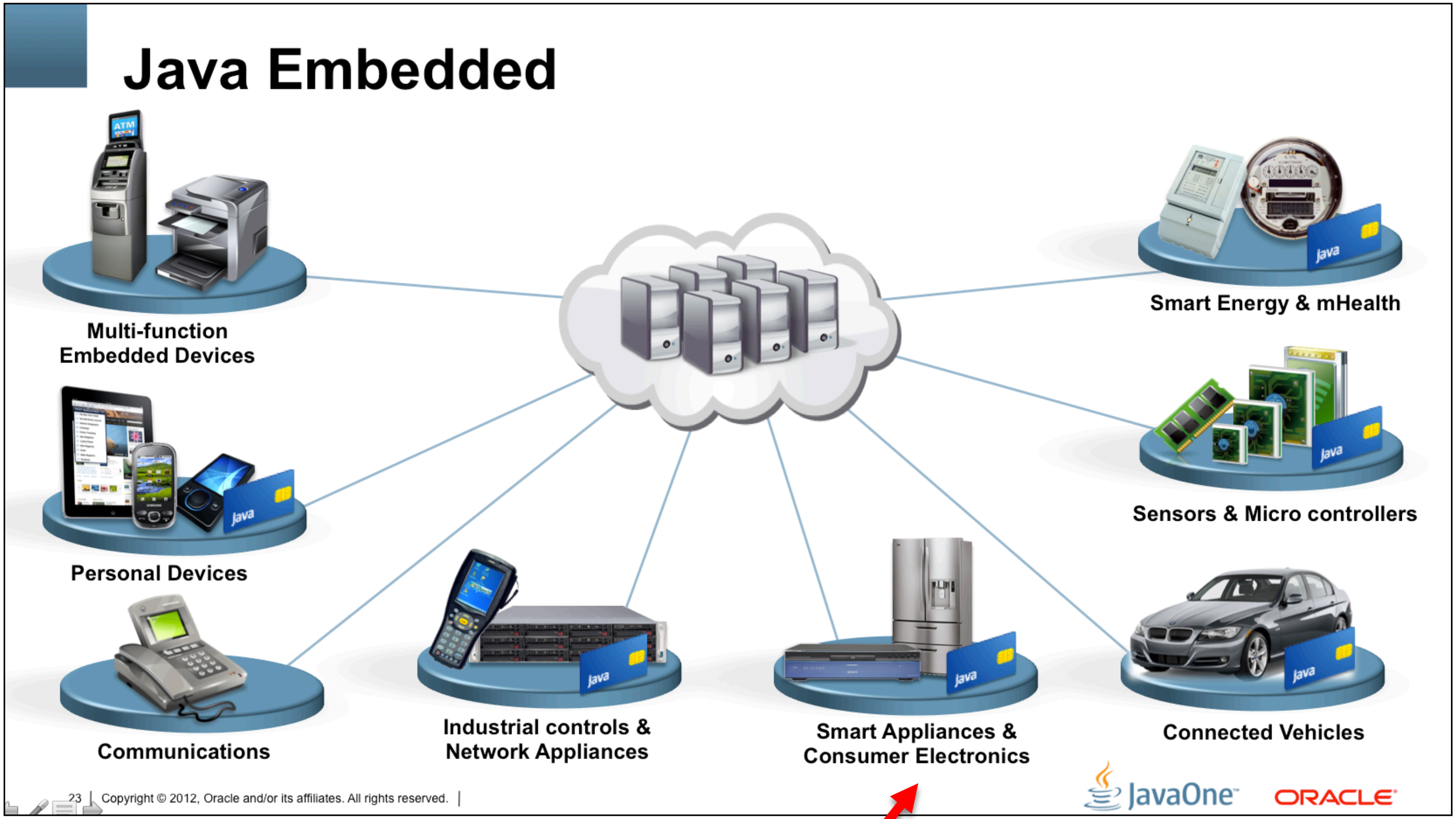
Major Changes
Project Jigsaw; designing and implementing a standard module system for the Java SE platform, and to apply that system to the platform itself and the the JDK.

PEARSON FRANK | JAVA, WEB & PHP RECRUITMENT EXPERTS

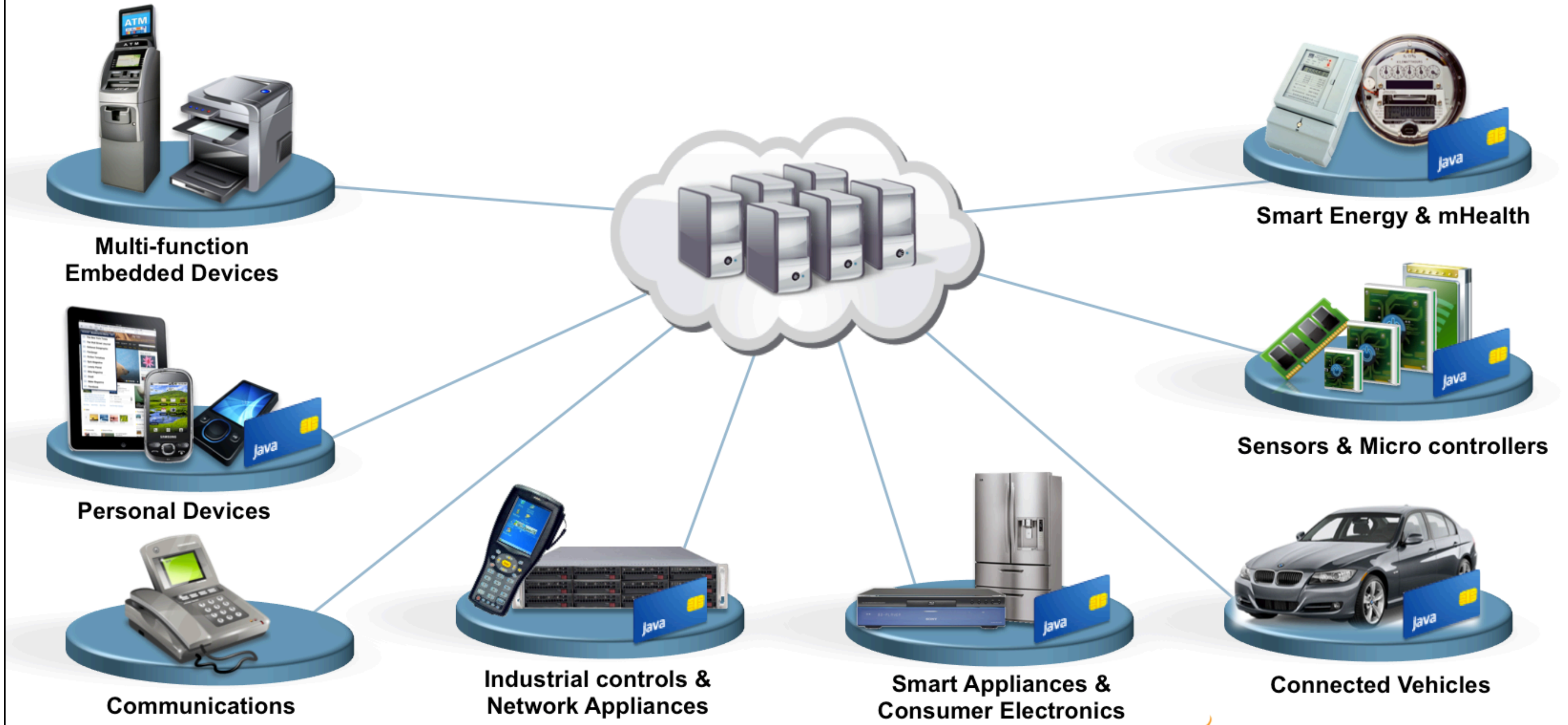
WWW.PEARSONFRANK.COM

Pearson Frank · @PearsonFrank_ · Pearson Frank

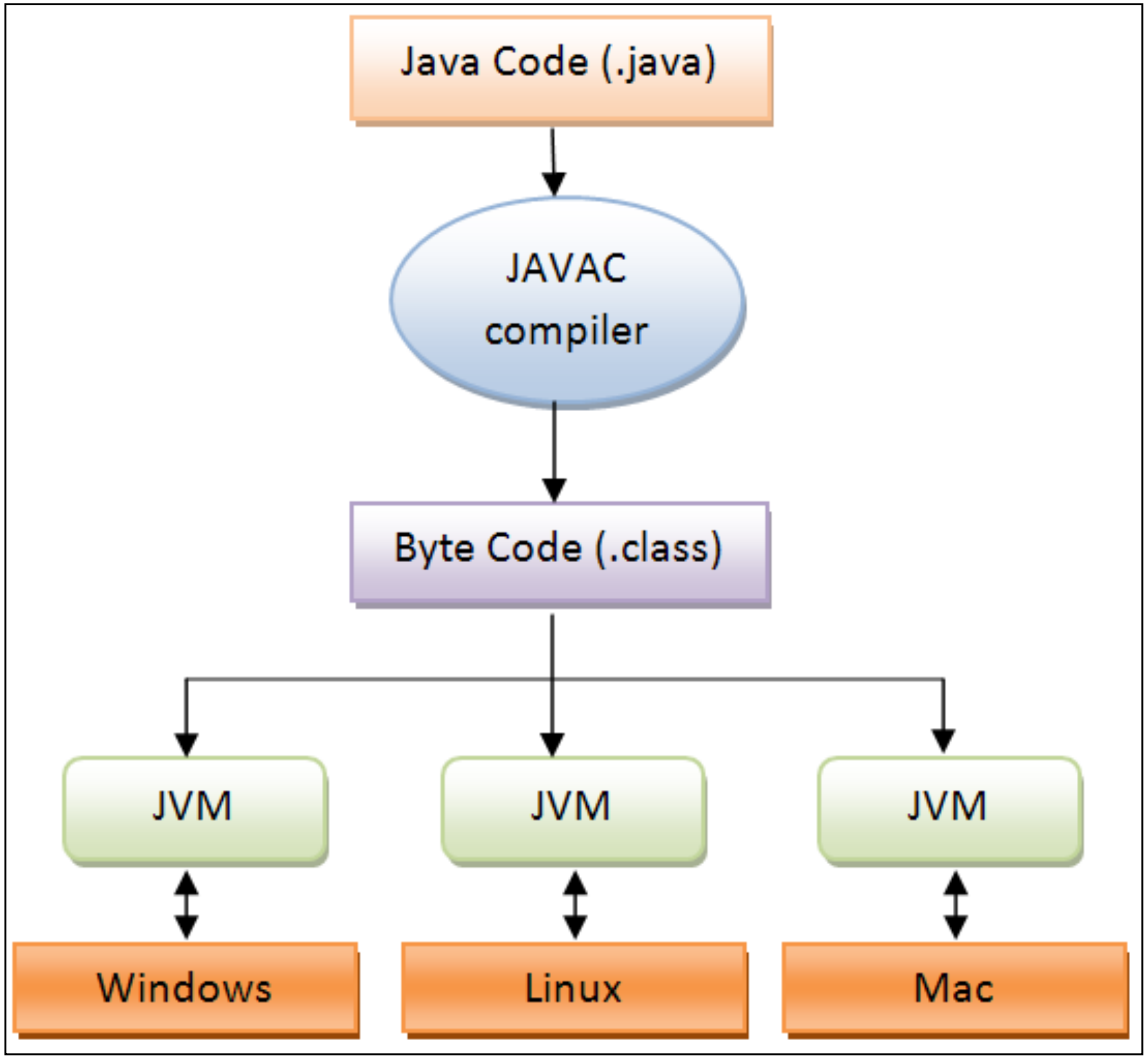
Initially intended for:



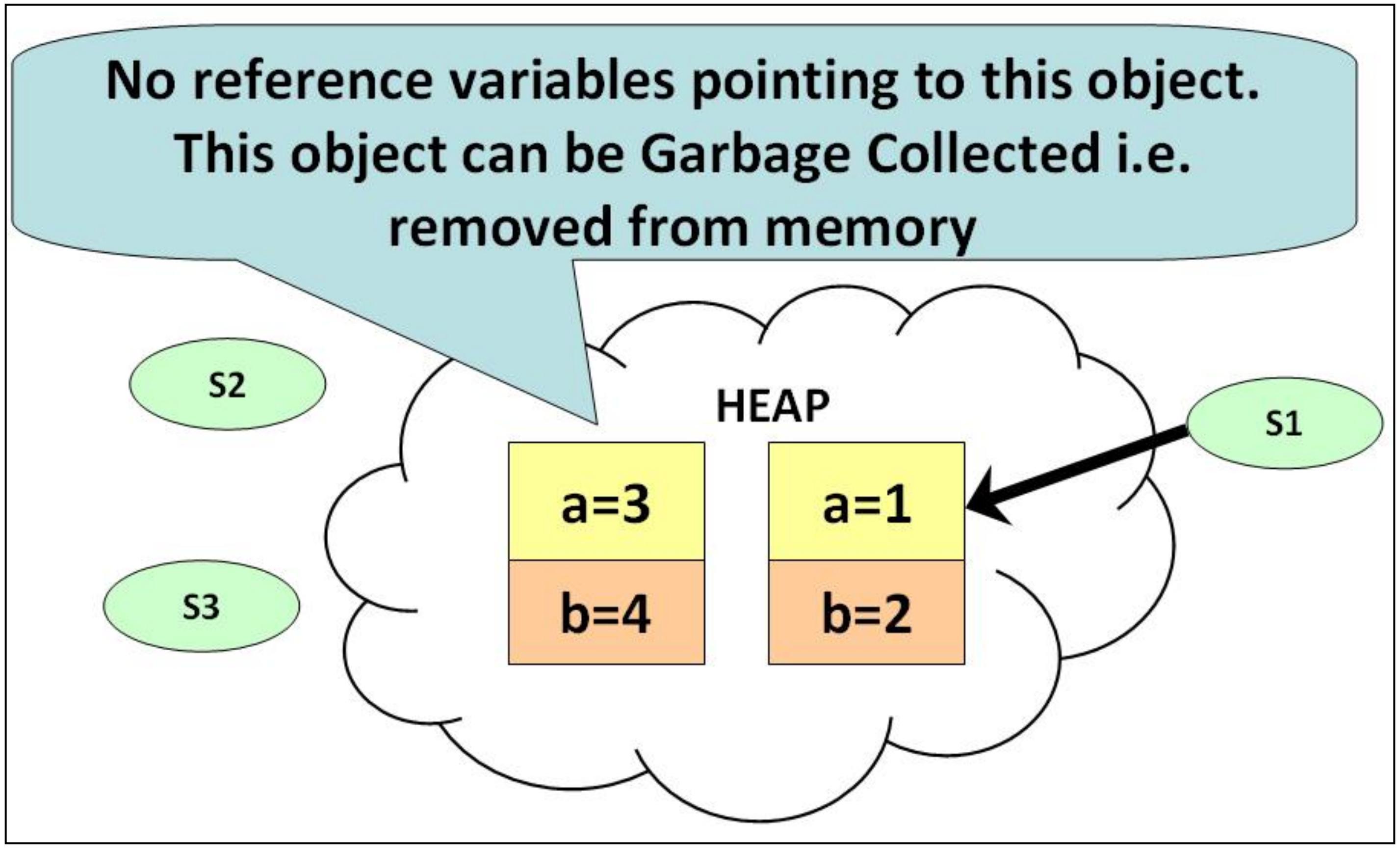
Java Embedded


Multi-function Embedded Devices
Personal Devices
Communications
Industrial controls & Network Appliances
Smart Appliances & Consumer Electronics
Connected Vehicles
Sensors & Micro controllers
Smart Energy & mHealth
java

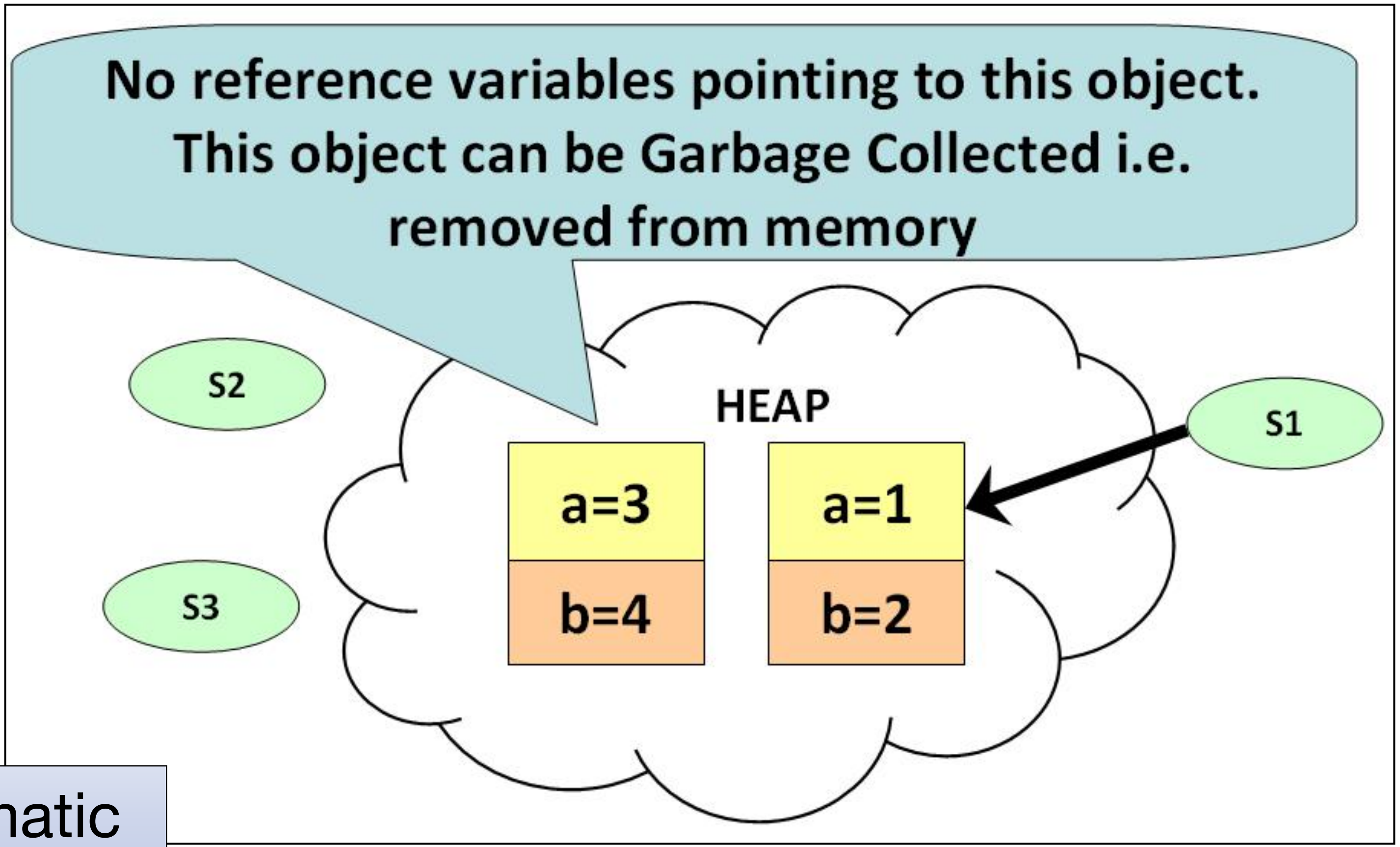
Portability / Compiler / JVM



Memory Management


No reference variables pointing to this object. This object can be Garbage Collected i.e. removed from memory
HEAP
S2
S3
S1
a=3
b=4
a=1
b=2

Memory Management


No reference variables pointing to this object. This object can be Garbage Collected i.e. removed from memory
S2
S3
HEAP
a=3
b=4
a=1
b=2
S1

Automatic

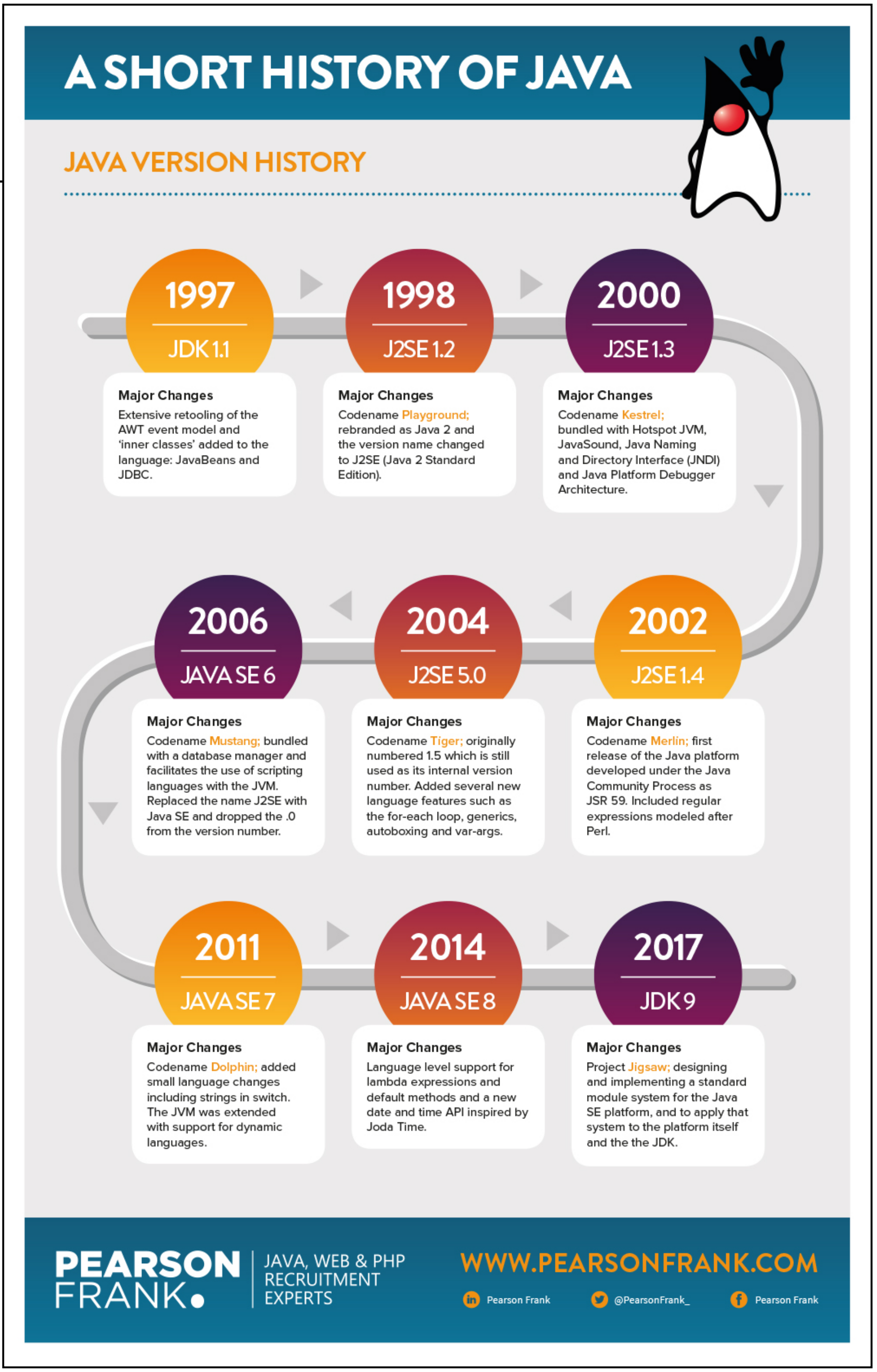
Happens when memory is required

Can be forced programmatically

Java Versions

Week(s)	Version
1 – 5	Focus on Java 7 constructs
6 +	Explore some of Java 8 and Java 9 changes

Java 9:
Currently available as an Early Access Download
→ Raw snapshots that let developers review and contribute to Java as it is being developed



A SHORT HISTORY OF JAVA

JAVA VERSION HISTORY

1997 — JDK 1.1
Major Changes
Extensive retooling of the AWT event model and 'inner classes' added to the language: JavaBeans and JDBC.

1998 — J2SE 1.2
Major Changes
Codename Playground; rebranded as Java 2 and the version name changed to J2SE (Java 2 Standard Edition).

2000 — J2SE 1.3
Major Changes
Codename Kestrel; bundled with Hotspot JVM, JavaSound, Java Naming and Directory Interface (JNDI) and Java Platform Debugger Architecture.

2002 — J2SE 1.4
Major Changes
Codename Merlin; first release of the Java platform developed under the Java Community Process as JSR 59. Included regular expressions modeled after Perl.

2004 — J2SE 5.0
Major Changes
Codename Tiger; originally numbered 1.5 which is still used as its internal version number. Added several new language features such as the for-each loop, generics, autoboxing and var-args.

2006 — JAVA SE 6
Major Changes
Codename Mustang; bundled with a database manager and facilitates the use of scripting languages with the JVM. Replaced the name J2SE with Java SE and dropped the .0 from the version number.

2011 — JAVA SE 7
Major Changes
Codename Dolphin; added small language changes including strings in switch. The JVM was extended with support for dynamic languages.

2014 — JAVA SE 8
Major Changes
Language level support for lambda expressions and default methods and a new date and time API inspired by Joda Time.

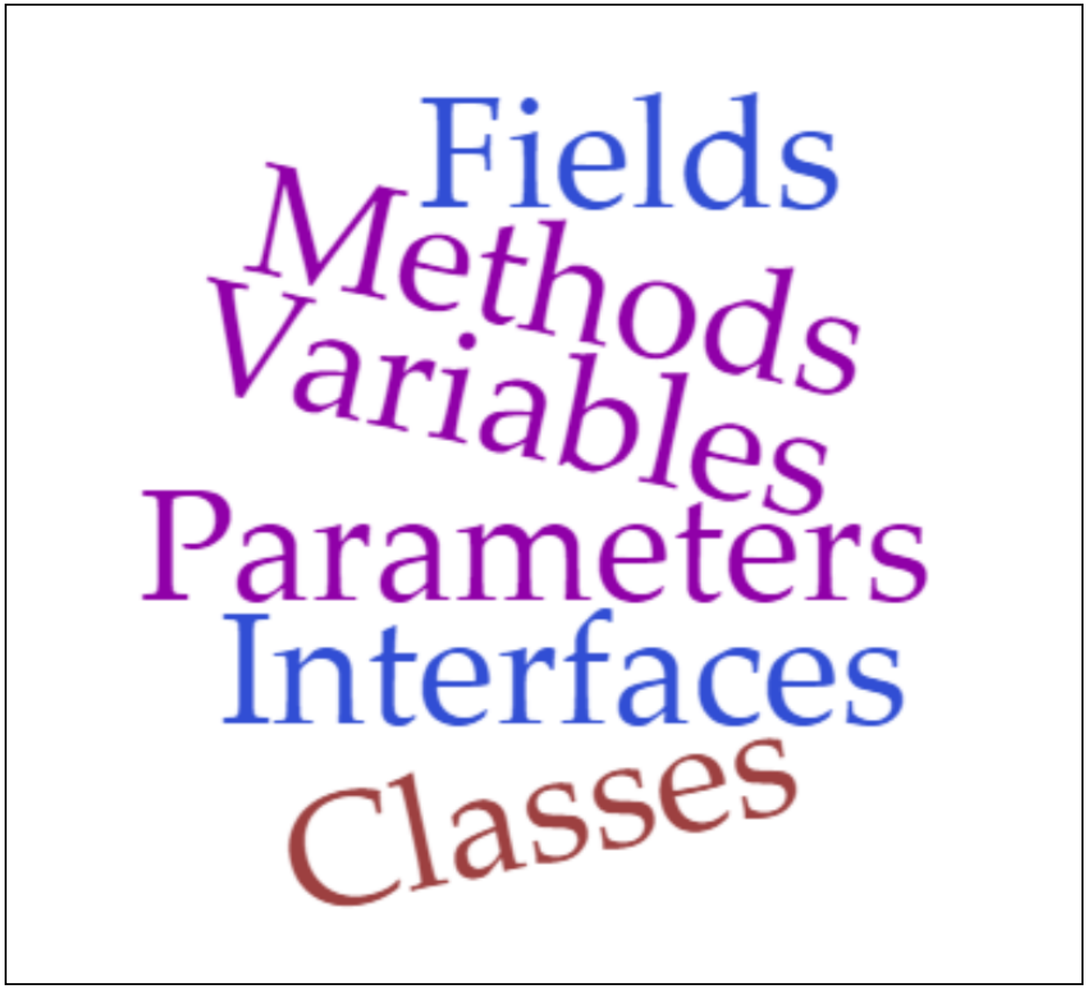
2017 — JDK 9
Major Changes
Project Jigsaw; designing and implementing a standard module system for the Java SE platform, and to apply that system to the platform itself and the the JDK.

PEARSON FRANK. | JAVA, WEB & PHP RECRUITMENT EXPERTS
WWW.PEARSONFRANK.COM
Pearson Frank @PearsonFrank_ Pearson Frank

Overview: Road Map

- Java Introduction
  - History
  - Portability
  - Compiler
  - Java Virtual Machine
  - Garbage collection
- Java Syntax
  - Identifiers
  - Expressions
  - Comments
- Java Basics
  - Java types
  - Primitives
  - Objects
  - Variables
  - Operators
  - Identity and equality
- Arrays
  - What are arrays?
  - Creating arrays
  - Using arrays

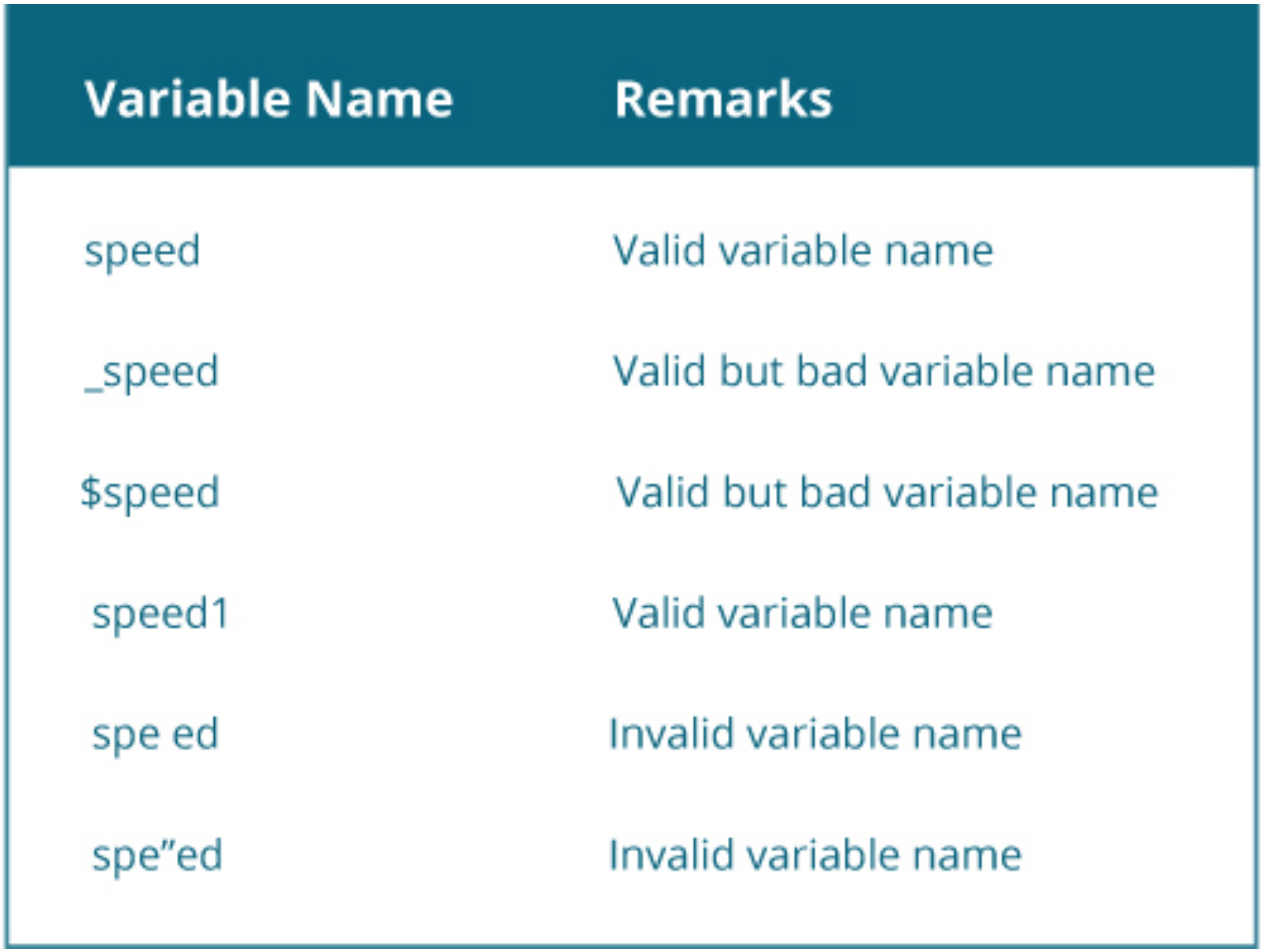
Identifiers are used for naming:



Identifiers

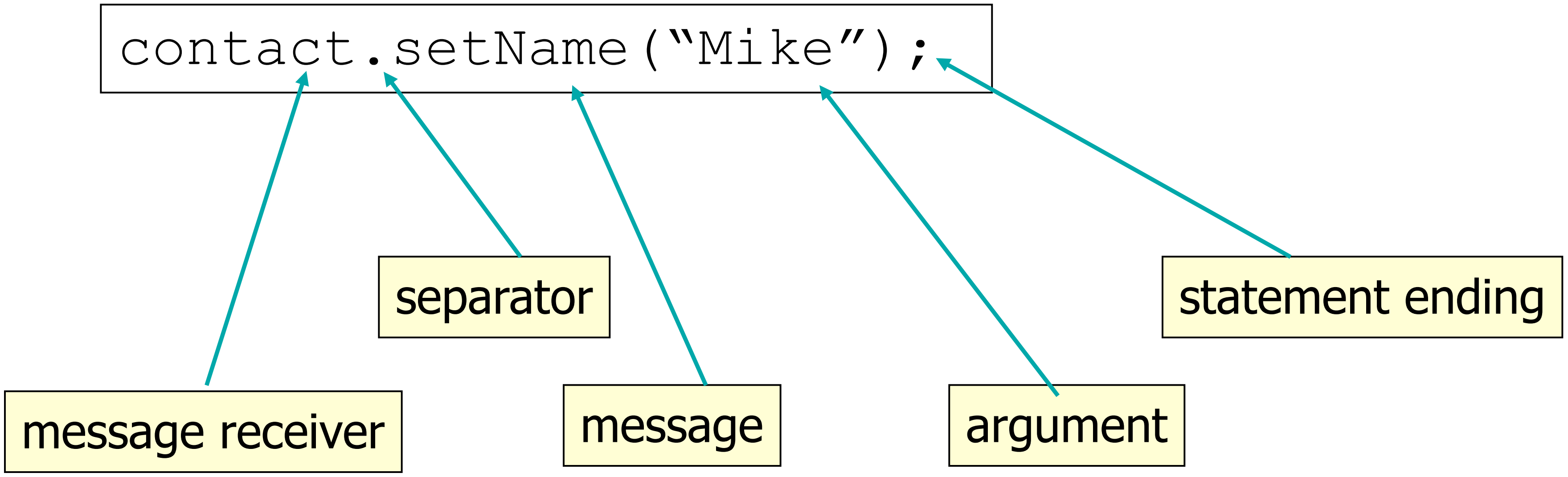
- Are case-sensitive.
- Begin with either:
    - a **letter (preferable)**,
    - the dollar sign "$", or
    - the underscore character "_".
- Can contain letters, digits, dollar signs, or underscore characters.
- Can be any length you choose.
- Must not be a **keyword or reserved word** e.g. int, while, etc.
- Cannot contain white spaces.

Identifiers



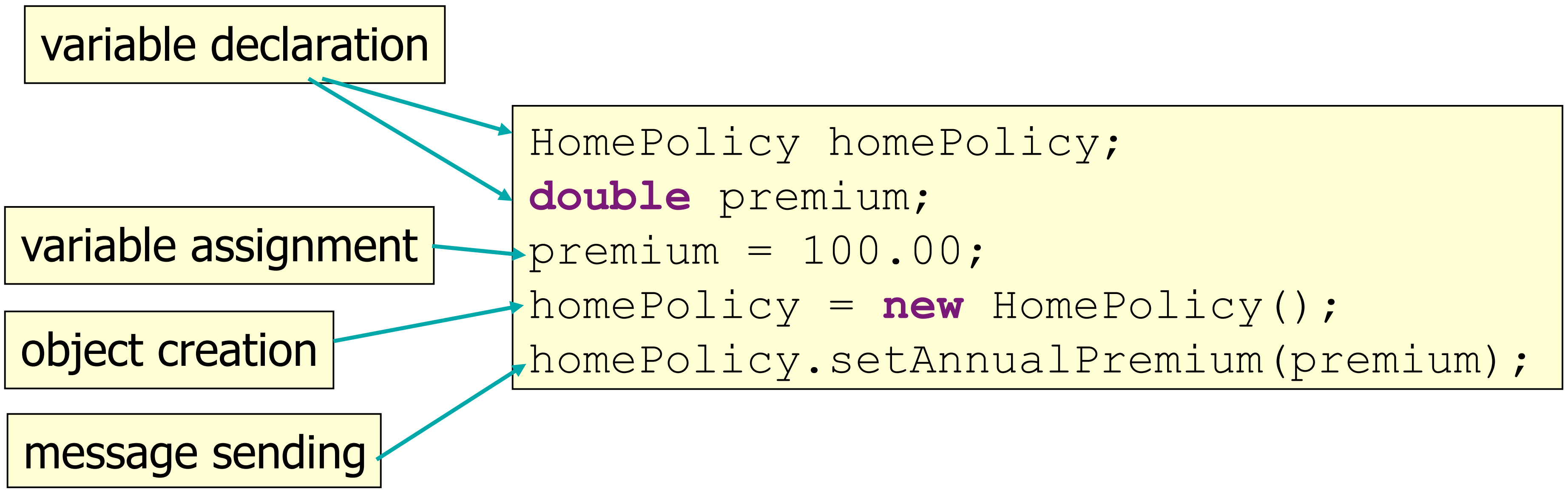
Variable Name	Remarks
speed	Valid variable name
_speed	Valid but bad variable name
$speed	Valid but bad variable name
speed1	Valid variable name
spe ed	Invalid variable name
spe"ed	Invalid variable name

Messages and Objects



```
contact.setName("Mike");
```

Statements → Basic Java Expressions



```
HomePolicy homePolicy;
double premium;
premium = 100.00;
homePolicy = new HomePolicy();
homePolicy.setAnnualPremium(premium);
```

Empty Expression

```
;      //this is an empty statement
       // on its own in the line
       //it means…do nothing!
```

```
for(int i=1; i<3; i++) ;
      System.out.println(i);
```

We would expect 1, 2 printed but it only prints 1 because of the statement terminator.

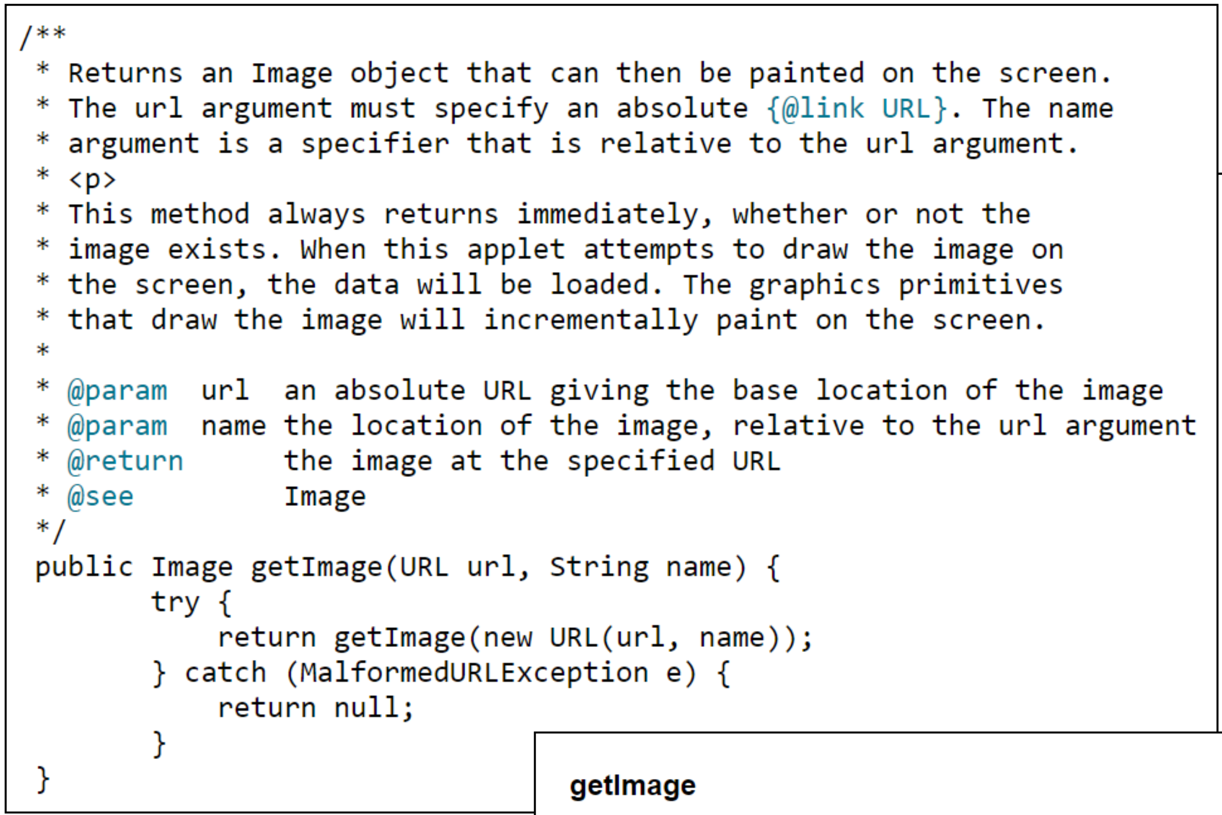
Comments

```
/** Javadoc example comment.
 *  Used for generation of the documentation.
 */

/*  Multiple line comment.
 *
 */

//  Single line comment.
```

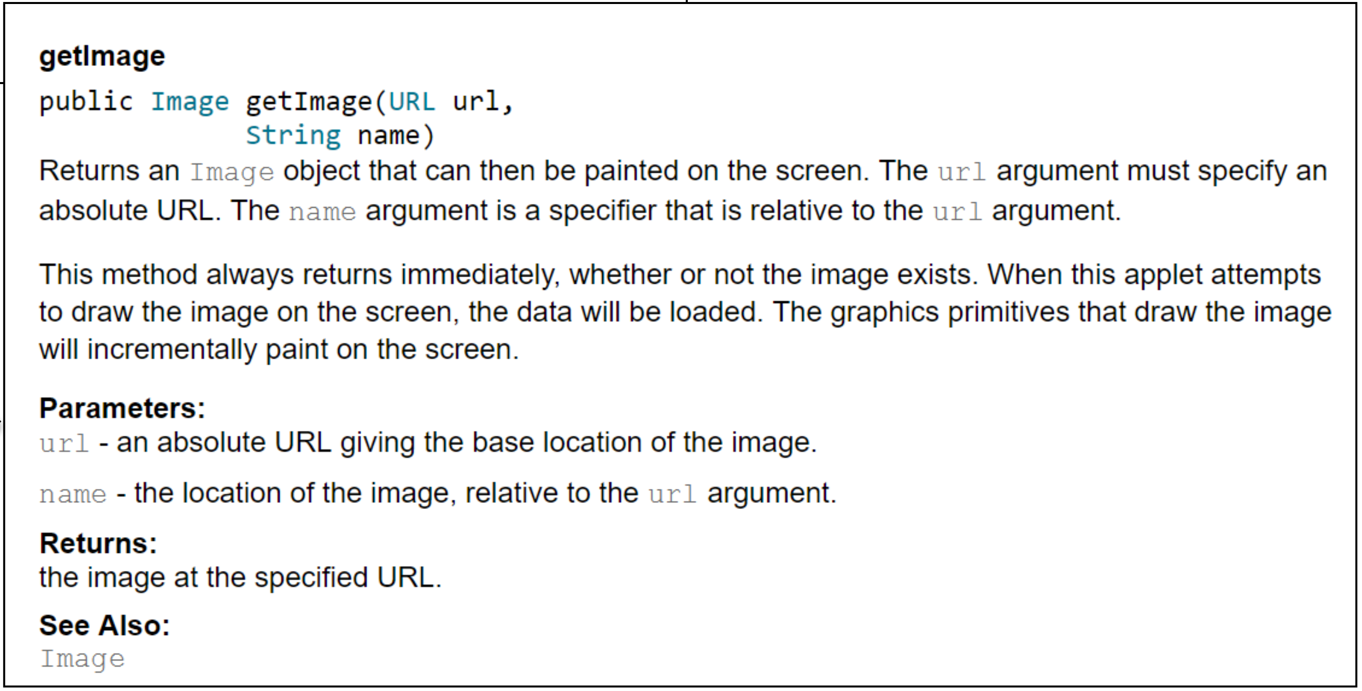
Javadoc



```
/**
 * Returns an Image object that can then be painted on the screen.
 * The url argument must specify an absolute {@link URL}. The name
 * argument is a specifier that is relative to the url argument.
 * <p>
 * This method always returns immediately, whether or not the
 * image exists. When this applet attempts to draw the image on
 * the screen, the data will be loaded. The graphics primitives
 * that draw the image will incrementally paint on the screen.
 *
 * @param  url  an absolute URL giving the base location of the image
 * @param  name the location of the image, relative to the url argument
 * @return      the image at the specified URL
 * @see         Image
 */
 public Image getImage(URL url, String name) {
        try {
            return getImage(new URL(url, name));
        } catch (MalformedURLException e) {
            return null;
        }
 }
```

Code in the Java file

Produces this HTML code



getImage

```
public Image getImage(URL url,
                      String name)
```

Returns an `Image` object that can then be painted on the screen. The `url` argument must specify an absolute URL. The `name` argument is a specifier that is relative to the `url` argument.

This method always returns immediately, whether or not the image exists. When this applet attempts to draw the image on the screen, the data will be loaded. The graphics primitives that draw the image will incrementally paint on the screen.

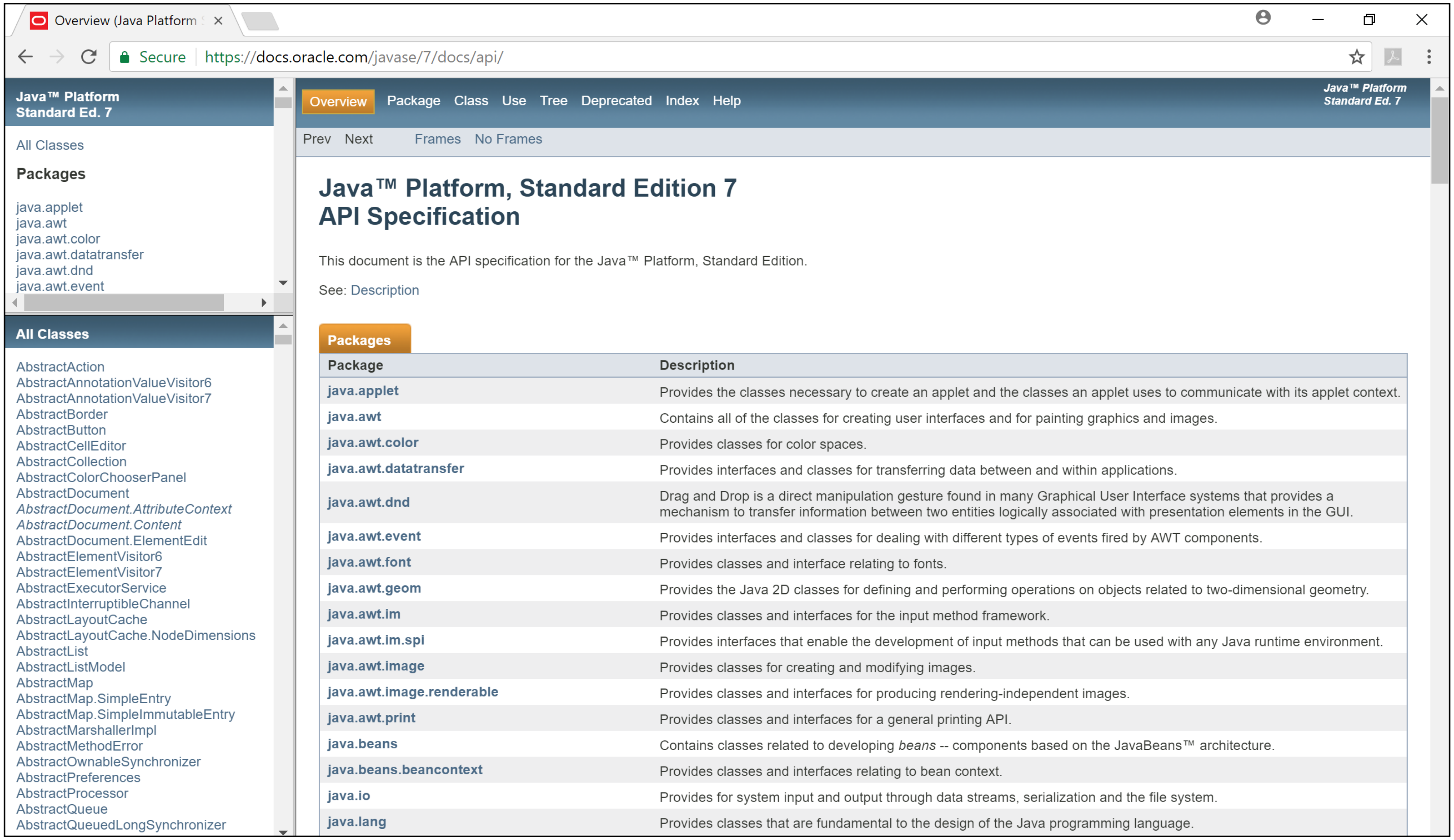
Parameters:
`url` - an absolute URL giving the base location of the image.

`name` - the location of the image, relative to the `url` argument.

Returns:
the image at the specified URL.

See Also:
`Image`

Java API → Javadoc output



Java™ Platform
Standard Ed. 7

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event

All Classes

AbstractAction
AbstractAnnotationValueVisitor6
AbstractAnnotationValueVisitor7
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeContext
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractElementVisitor6
AbstractElementVisitor7
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDimensions
AbstractList
AbstractListModel
AbstractMap
AbstractMap.SimpleEntry
AbstractMap.SimpleImmutableEntry
AbstractMarshallerImpl
AbstractMethodError
AbstractOwnableSynchronizer
AbstractPreferences
AbstractProcessor
AbstractQueue
AbstractQueuedLongSynchronizer

Overview Package Class Use Tree Deprecated Index Help

Java™ Platform
Standard Ed. 7

Prev Next Frames No Frames

**Java™ Platform, Standard Edition 7
API Specification**

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing *beans* -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.

Literals

```
String one = "One";
String two = "Two";
```

=

```
String one = new String("One");
String two = new String("Two");
```

What we covered in this lecture:

- **Overview**
  - Introduction
  - Syntax
  - Basics
  - Arrays
- **Classes**
  - Classes Structure
  - Static Members
  - Commonly used Classes
- **Control Statements**
  - Control Statement Types
  - If, else, switch
  - For, while, do-while

- **Inheritance**
  - Class hierarchies
  - Method lookup in Java
  - Use of this and super
  - Constructors and inheritance
  - Abstract classes and methods

  Interfaces
- **Collections**
  - ArrayList
  - HashMap
  - Iterator
  - Vector
  - Enumeration
  - Hashtable

- **Exceptions**
  - Exception types
  - Exception Hierarchy
  - Catching exceptions
  - Throwing exceptions
  - Defining exceptions

  Common exceptions and errors
- **Streams**
  - Stream types
  - Character streams
  - Byte streams
  - Filter streams
  - Object Serialization