1. What is Java?

Thanks to Java is a new programming language developed at Sun under the direction of James Gosling. As far as possible it is based on concepts from C, Objective C and C++.

Java is interpreted and loads classes dynamically. There are CPU chips for Java; Sun showed a prototype Java computer early in 1996 and by now it is commercially available (if slow).

HotJava is a Web browser, that was implemented in Java using a modular architecture. It loads protocol modules and other applications (applets) dynamically from the local system or over a network and thus adapts to new tasks.

According to Hoff, Shaio and Starbuck, Java is “simple, object oriented, statically typed, compiled, architecture neutral, multi-threaded, garbage collected, robust, secure, extensible and well understood. Above all, however, Java is fun!”

These terms are elaborated on below -- I do not quite accept all these claims, however ...

simple

Java is related to languages like C or C++, but to make the language small and easy to learn all inessentials were eliminated. This leads to more uniform programs and simpler maintenance.

Unfortunately, Java 1.1 introduced significant new, useful, but different concepts. There is a proliferation of libraries as various companies try to turn individuality into market dominance.
object-oriented
Modern programs are distributed and have graphical user interfaces. Both can be implemented more easily with object orientation. Unlike C++, Java is fully object oriented and thus furthers the right programming style for these problem areas.

statically typed
All data must be declared with types so that data and operations may be matched as far as possible during compilation. Methods are dynamically bound but overloaded signatures are decided during compilation. Like Objective C, Java permits type queries and object analysis, e.g., for the existence of methods, using the package java/lang/reflect at runtime.

compiled
Unlike TCL, Java avoids repeated source analysis. A program is compiled into byte codes, the machine language of the Java Virtual Machine (JVM). The JVM is interpreted or compiled just in time. Classes and methods are bound symbolically, i.e., classes can be recompiled individually.

architecture neutral
Byte codes are the same everywhere; only the JVM has to be implemented for a new platform. Java programs should run everywhere and can be distributed as binaries. Unlike C and C++, Java completely specifies the capacity and behavior of the primitive data types thus eliminating a serious portability problem. Swing is implemented without any native code, relying only on the API defined in JDK 1.1.
multi-threaded
Graphical user interfaces provide the illusion of parallel execution. Threads offer an elegant implementation. Java has a thread system based on classes and the language contains simple synchronization mechanisms (monitors). Many class libraries are thread-safe.

garbage collected
Dynamic memory management as in C and C++, where the programmer attends to reusing resources, is efficient but error prone. Java only knows dynamic objects and vectors and completely frees the programmer from the problem of memory reuse. Garbage collection runs as a parallel thread and thus should not be a bottleneck in critical situations.

robust
Exceptions are an integral part of Java for error handling. The programmer is constantly forced to consider error possibilities in libraries and the compiler can check that exceptions are not hidden or overlooked.

secure
An interpreter can pretty much ensure that the interpreted program cannot crash it’s platform. In connection with the Web, Java has additional security mechanisms that constrain foreign programs so that viruses are considered impossible — in spite of the fact that binary Java programs can run on arbitrary platforms.
Various groups have demonstrated, however, that security holes do exist. There is a move toward digitally signed programs and distributed trust.

extensible
Java methods can be implemented in other languages using the Java Native Interface (JNI). In principle, arbitrary libraries can be accessed as long as other security or portability aspects do not prevail.

well understood
While Java is a new language it’s concepts are well known. The language definition is comprehensive and still short. Unlike C++, a programmer can certainly understand Java completely and use all of it. Java’s relationship to C and it’s dialects makes it easy to get into the language, although there are a number of subtle differences to be aware of.
1.1. Different JVMS ...

Java Standard Edition (J2SE)
Java 2 Micro Edition (J2ME), Connected Device Configuration (CDC)
Java 2 Micro Edition (J2ME), Connected Limited Device Configuration (CLDC)

<table>
<thead>
<tr>
<th>Edition</th>
<th>Configuration</th>
<th>RAM</th>
<th>Typical Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2ME</td>
<td>CLDC</td>
<td>32k</td>
<td>Cellphone, pagers</td>
</tr>
<tr>
<td>J2ME</td>
<td>CDC</td>
<td>512k</td>
<td>PDA's</td>
</tr>
<tr>
<td>J2SE</td>
<td>CDC</td>
<td>Lots</td>
<td>Desktop applications</td>
</tr>
</tbody>
</table>

1.2. J2SE Platform at a Glance

Copied from

1.3. The first Program: Hello.java

```java
/**
 * Classical: Hello, World
 * @version $Id: Hello.java,v 1.3 2001/06/06 23:05:46 hpb Exp hpb $
 * @author hpb
 * Revisions:
 * Revision 1.41 2013/06/06 16:19:12 hpb
 * Revision 1.42 2014/08/06 10:31:21 hpb
 * Initial revision
 *
 class Hello {
     public static void main (String args []) { // main program
         System.out.println("Hello World!");
     }
 }
```

Source Code: Src/3/Hello.java

% javac Hello.java
% java Hello
Hello, World

Please take a look at the
Please take a look into how to use
1.4. The first Application

Hello outputs a text that is specified in the program.

Hello demonstrates how to code a minimal application.

An application is a stand-alone Java program with a graphical user interface, usually based on the Abstract Window Toolkit

```java
import javax.swing.*;
import java.awt.event.*; // for WindowEvent

public class HelloWorld {

    public static void main(String[] args) {
        JFrame frame = new JFrame("HelloWorld");
        final JLabel label = new JLabel("Hello World");
        frame.getContentPane().add(label);

        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });

        frame.pack();
        frame.setVisible(true);
    }
}

Source Code: Src/3/HelloWorld.java
```
1.5. The first applet -- applets/hello

Hello outputs a text that is specified in the program.

Hello is an applet and demonstrates how Java is used in Web documents.

```java
/**
 * Classical Applet: Hello, World
 * @version $Id$
 * @author hpb
 * Revisions:
 * $Log$
 */
import java.applet.*; // find Applet class
import java.awt.*; // find Graphics class
import javax.swing.*; // find Swing class

/** A class with the first Java applet */
public class HelloApplet extends JApplet {

    /** hook to initialize the applet */
    public void init () {
        resize(150, 25);
    }

    /** redraw the Applet */
    public void paint (Graphics g) {
        g.drawString("Hello, World", 50, 25);
    }
}
```

Source Code: Src/3/HelloApplet.java
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
</table>
| 21   | `<font color="#3f5fbf">/**</font>`<br/>| 22   | `<font color="#ffffff">&nbsp;</font><font color="#3f5fbf">Classical:&nbsp;Hello,&nbsp;</font>`<br/>| 23   | `<font color="#ffffff">&nbsp;</font><font color="#3f5fbf">*&nbsp;</font>`<br/>| 24   | `<font color="#ffffff">&nbsp;</font><font color="#3f5fbf">*&nbsp;</font>`<br/>| 25   | `<font color="#ffffff">&nbsp;</font><font color="#3f5fbf">*&nbsp;</font>`<br/>| 26   | `<font color="#ffffff">&nbsp;</font><font color="#3f5fbf">*&nbsp;</font>`<br/>| 27   | `<font color="#ffffff">&nbsp;</font><font color="#3f5fbf">*&nbsp;Revision&nbsp;1.1&nbsp;2003/06/06&nbsp;16:19:12&nbsp;hpb</font>`<br/>| 28   | `<font color="#ffffff">&nbsp;</font><font color="#3f5fbf">*&nbsp;Initial&nbsp;revision</font>`<br/>| 29   | `<font color="#ffffff">&nbsp;</font><font color="#3f5fbf">*</font>`<br/>| 30   | `<font color="#ffffff">&nbsp;</font><font color="#3f5fbf">*/</font>`<br/>| 31   | `<font color="#7f0055"><b>class&nbsp;</b></font><font color="#000000">Hello&nbsp;</font>`<br/>| 32   | `<font color="#000000">{</font>`<br/>| 33   | `<font color="#ffffff">&nbsp;&nbsp;&nbsp;&nbsp;</font><font color="#7f0055"><b>public&nbsp;static&nbsp;</b></font><font color="#000000">main</font>`<br/>| 34   | `<font color="#000000">(</font>`<br/>| 35   | `<font color="#000000">String&nbsp;</font>`<br/>| 36   | `<font color="#000000">System.out.println</font>`<br/>| 37   | `<font color="#000000">("Hello&nbsp;World!")</font>`<br/>| 38   | `<font color="#000000">);</font>`<br/>| 39   | `<font color="#000000">}</font>`

Source Code: Src/3/Hello.html

% appletviewer Hello.html

gives us:

% netscape Hello.html

gives us an error message because of using swing. We will solve this problem later.
1.6. Documentation — javadoc

The various JDK packages are documented on HTML pages in a standardized format that contains very many references to other classes or replaced methods etc.

See here:

creates the documentation directly from the program sources. Special comments /** ... */ before class, variable and method declarations are transferred to the documentation. The comments may contain significant tags

See coding standard