Handout chapter 1

a)

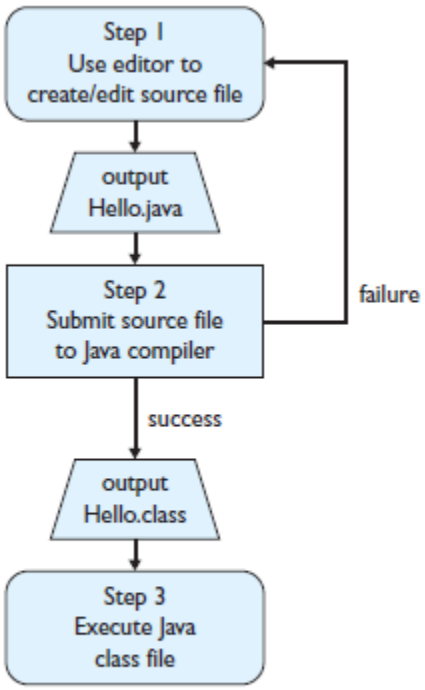

Figure 1.1 Creation and execution of a Java program

b)

```
1  public class Hello2 {
2      public static void main(String[] args) {
3          System.out.println("Hello, world!");
4          System.out.println();
5          System.out.println("This program produces four");
6          System.out.println("lines of output.");
7      }
8  }
```

c)

It may seem odd to put the opening curly brace at the end of a line rather than on a line by itself. Some people would use this style of indentation for the program instead:

```
1  public class Hello3
2  {
3      public static void main(String[] args)
4      {
5          System.out.println("Hello, world!");
6      }
7  }
```

Different people will make different choices about the placement of curly braces.

d)

```
System.out.println("This is the first line of output.");
System.out.println();
System.out.println("This is the third, below a blank line.");
```

Executing these statements produces the following three lines of output (the second line is blank):

```
This is the first line of output.

This is the third, below a blank line.
```

e)

The solution is to embed what are known as *escape sequences* in the string literals. Escape sequences are two-character sequences that are used to represent special characters. They all begin with the backslash character (\). Table 1.3 lists some of the more common escape sequences.

Table 1.3 Common Escape Sequences

Sequence	Represents
\t	tab character
\n	new line character
\"	quotation mark
\\	backslash character

Keep in mind that each of these two-character sequences actually stands for just a single character. For example, consider the following statement:

```
System.out.println("What \"characters\" does this \\ print?");
```

If you executed this statement, you would get the following output:

```
What "characters" does this \ print?
```

f)

- `AllMyChildren` for a class name (each word starts with a capital)
- `allMyChildren` for a method name (starts with a lowercase letter, subsequent words capitalized)
- `ALL_MY_CHILDREN` for a constant name (all uppercase, with words separated by underscores; described in Chapter 2)

Java is case sensitive, so the identifiers `class`, `Class`, `CLASS`, and `cLASs` are all considered different.

G)

Table 1.4 List of Java Keywords

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

H)

Java is a free-format language. This means you can put in as many or as few spaces and blank lines as you like, as long as you put at least one space or other punctuation mark between words. However, you should bear in mind that the layout of a program can enhance (or detract from) its readability. The following program is legal but hard to read:

```
1  public class Ugly{public static void main(String[] args)
2  {System.out.println("How short I am!");}}
```

Using these rules to rewrite the `Ugly` program yields the following code:

```
1  public class Ugly {
2      public static void main(String[] args) {
3          System.out.println("How short I am!");
4      }
5  }
```

I)

Common Programming Error

File Name Does Not Match Class Name

As mentioned earlier, Java requires that a program's class name and file name match. For example, a program that begins with `public class Hello` must be stored in a file called `Hello.java`.

If you use the wrong file name (for example, saving it as `WrongFileName.java`), you'll get an error message like this:

```
WrongFileName.java:1: class Hello is public, should be
    declared in a file named Hello.java
public class Hello {
       ^
```

Common Programming Error

Misspelled Words

Java (like most programming languages) is very picky about spelling. You need to spell each word correctly, including proper capitalization. Suppose, for example, that you were to replace the `println` statement in the "hello world" program with the following:

```
System.out.pruntln("Hello, world!");
```

When you try to compile this program, it will generate an error message similar to the following:

```
Hello.java:3: cannot find symbol
symbol  : method pruntln(java.lang.String)
```

Common Programming Error

Forgetting a Semicolon

All Java statements must end with semicolons, but it's easy to forget to put a semicolon at the end of a statement, as in the following program:

```
1  public class MissingSemicolon {
2      public static void main(String[] args) {
3          System.out.println("A rose by any other name")
```

Common Programming Error

Forgetting a Required Keyword

Another common syntax error is to forget a required keyword when you are typing your program, such as `static` or `class`. Double-check your programs against the examples in the textbook to make sure you haven't omitted an important keyword.

The compiler will give different error messages depending on which keyword is missing, but the messages can be hard to understand. For example, you might write a program called `Bug4` and forget the keyword `class` when writing its class header. In this case, the compiler will provide the following error message:

```
Bug4.java:1: class, interface, or enum expected
public Bug4 {
       ^
1 error
```

However, if you forget the keyword `void` when declaring the main method, the compiler generates a different error message:

```
Bug5.java:2: invalid method declaration; return type required
    public static main(String[] args) {
                  ^
1 error
```

J)

```
1  public class DrawBoxes {
2       public static void main(String[] args) {
3           System.out.println("+------+");
4           System.out.println("|      |");
5           System.out.println("|      |");
6           System.out.println("+------+");
7           System.out.println();
8           System.out.println("+------+");
9           System.out.println("|      |");
10          System.out.println("|      |");
11          System.out.println("+------+");
12      }
13   }
```

```
1  public class DrawBoxes2 {
2      public static void main(String[] args) {
3          drawBox();
4          System.out.println();
5          drawBox();
6      }
7
8      public static void drawBox() {
9          System.out.println("+------+");
10         System.out.println("|      |");
11         System.out.println("|      |");
12         System.out.println("+------+");
13     }
14 }
```

L)

```
1  public class FooBarBazMumble {
2      public static void main(String[] args) {
3          foo();
4          bar();
5          System.out.println("mumble");
6      }
7
8      public static void foo() {
9          System.out.println("foo");
10     }

11
12     public static void bar() {
13         baz();
14         System.out.println("bar");
15     }
16
17     public static void baz() {
18         System.out.println("baz");
19     }
20 }
```

M)

```
1  public class Infinite {
2      public static void main(String[] args) {
3          oops();
4      }
5
6      public static void oops() {
7          System.out.println("Make it stop!");
8          oops();
9      }
10 }
```