

JAVA™

AN INTRODUCTION TO
PROBLEM SOLVING
AND PROGRAMMING

7TH EDITION

WALTER SAVITCH

Introduction to Computers and Java

1

FIGURE 1.1 Main Memory

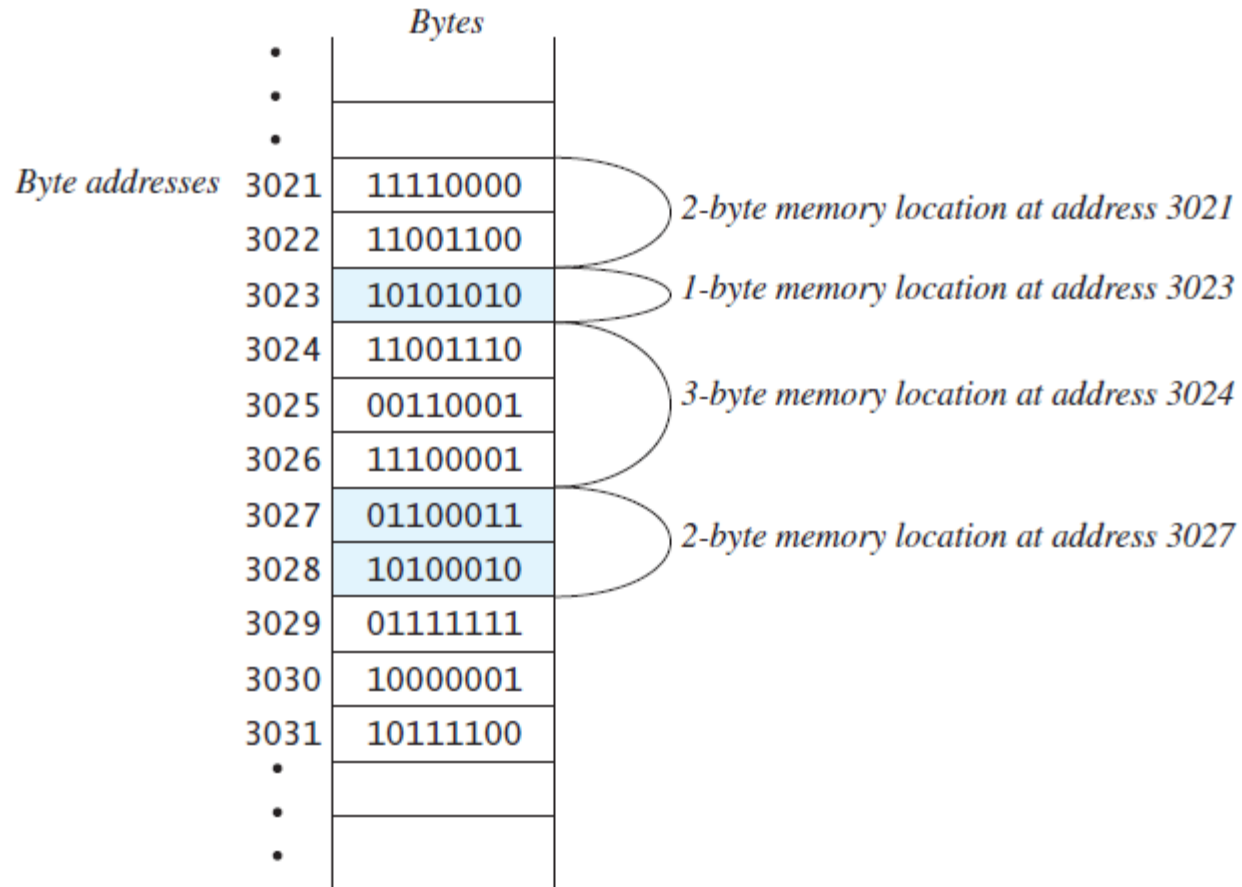


FIGURE 1.2 Running a Program

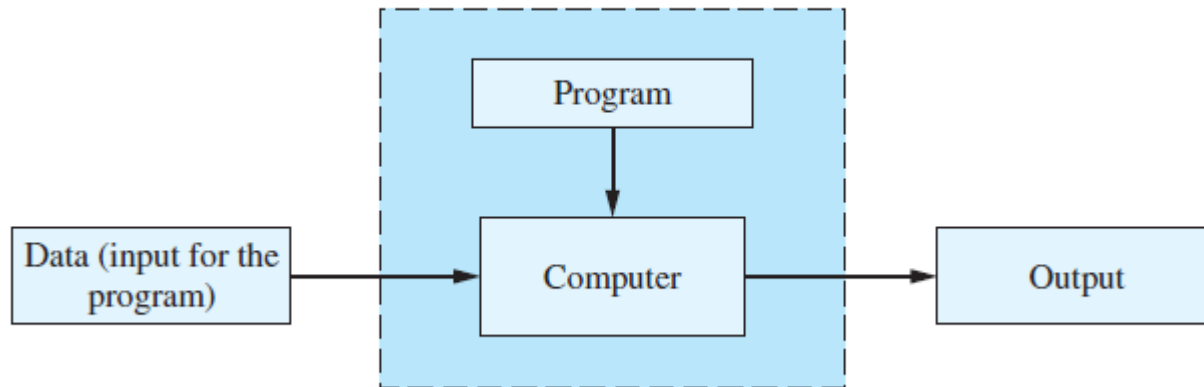
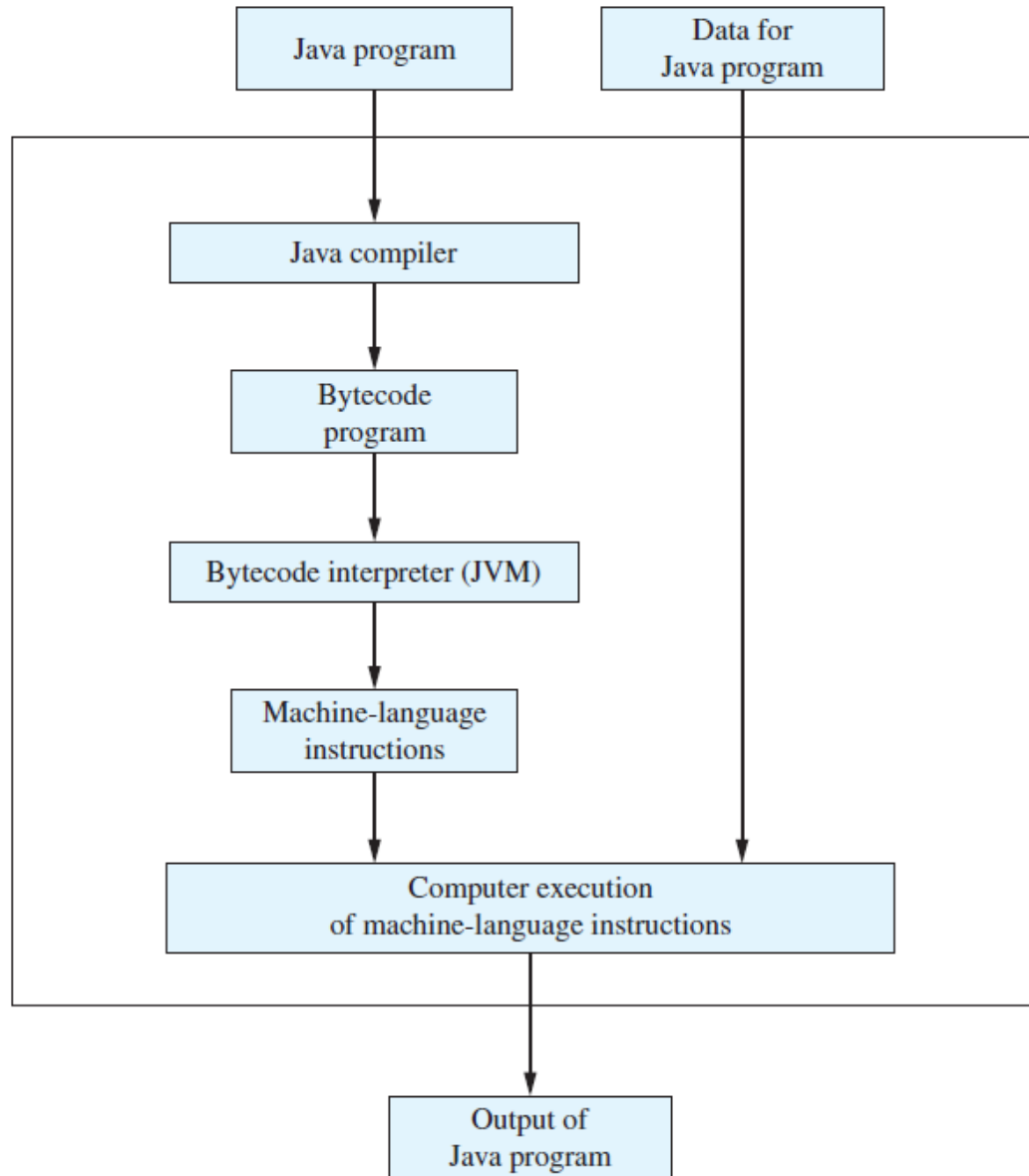


FIGURE 1.3 Compiling and Running a Java Program



LISTING 1.1 A Sample Java Program

```
import java.util.Scanner;
public class FirstProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello out there.");
        System.out.println("I will add two numbers for you.");
        System.out.println("Enter two whole numbers on a line:");

        int n1, n2;

        Scanner keyboard = new Scanner(System.in);

        n1 = keyboard.nextInt();
        n2 = keyboard.nextInt();

        System.out.println("The sum of those two numbers is");
        System.out.println(n1 + n2);
    }
}
```

Gets the Scanner class from the package (library) java.util

Name of the class—your choice

Sends output to screen

Says that n1 and n2 are variables that hold integers (whole numbers)

Reads the program for keyboard input

Reads one whole number from the keyboard

Sample Screen Output

```
Hello out there.
I will add two numbers for you.
Enter two whole numbers on a line:
12 30
The sum of those two numbers is
42
```

FIGURE 1.4 An Inheritance Hierarchy

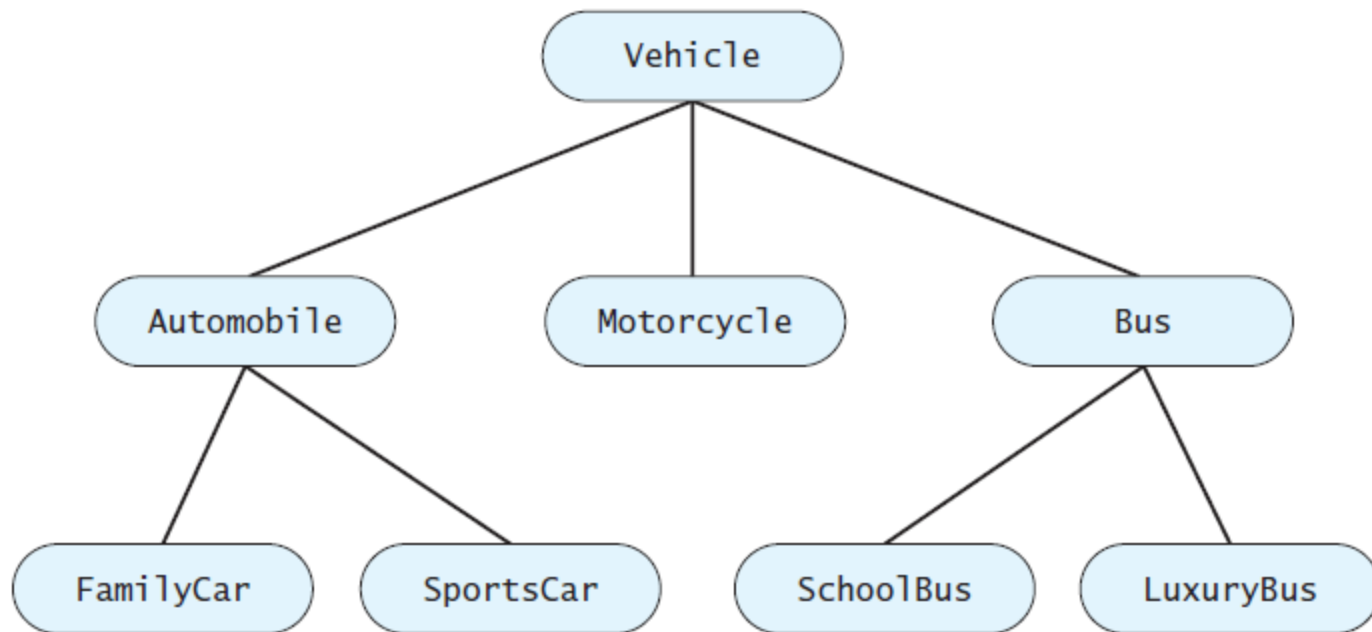


FIGURE 1.5 The Documentation for the Class Scanner

The screenshot shows the Java Platform Standard Ed. 7 API documentation for the `Scanner` class. The left sidebar contains a tree of package and class names. The main content area displays the class details for `Scanner`.

Package names (pointing to the left sidebar):

- java.applet
- java.awt
- java.awt.dnd
- java.awt.image
- java.beans
- java.io
- java.lang
- java.lang.annotation
- java.lang.invoke
- java.lang.management
- java.lang.module
- java.lang.ref
- java.lang.reflect
- java.lang.runtime
- java.lang.security
- java.lang.util
- java.net
- java.nio
- java.nio.channels
- java.nio.charset
- java.nio.file
- java.rmi
- java.security
- java.security.cert
- java.security.interfaces
- java.sql
- java.text
- java.time
- java.util
- java.util.concurrent
- java.util.concurrent.atomic
- java.util.concurrent.locks
- java.util.logging
- java.util.regex
- java.util.zip

Class names (we clicked on Scanner) (pointing to the left sidebar):

- Scanner
- ScatterlingByteChan
- ScheduledExecutor
- ScheduledFuture
- ScheduledThreadPo
- Schema
- SchemaFactory
- SchemaFactoryLoa
- SchemaOutputReso
- SchemaViolationExc
- ScriptContext
- ScriptEngine

Description of the class Scanner (pointing to the main content area):

Please note that this documentation is not final and is subject to change.

Overview Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS

SUMMARY: NESTED | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

Java™ Platform Standard Ed. 7 DRAFT ea-b97

java.util

Class Scanner

java.lang.Object

↳ **java.util.Scanner**

All Implemented Interfaces:

Iterator<String>

public final class Scanner

extends **Object**

implements **Iterator<String>**

A simple text scanner which can parse primitive types and strings using regular expressions.

A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various next methods.

For example, this code allows a user to read a number from `System.in`:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

LISTING 1.2 Drawing a Happy Face

```
import javax.swing.JApplet;  
import java.awt.Graphics;  
public class HappyFace extends JApplet  
{  
    public void paint(Graphics canvas)  
    {  
        canvas.drawOval(100, 50, 200, 200);  
        canvas.fillOval(155, 100, 10, 20);  
        canvas.fillOval(230, 100, 10, 20);  
        canvas.drawArc(150, 160, 100, 50, 180, 180);  
    }  
}
```

Applet Output

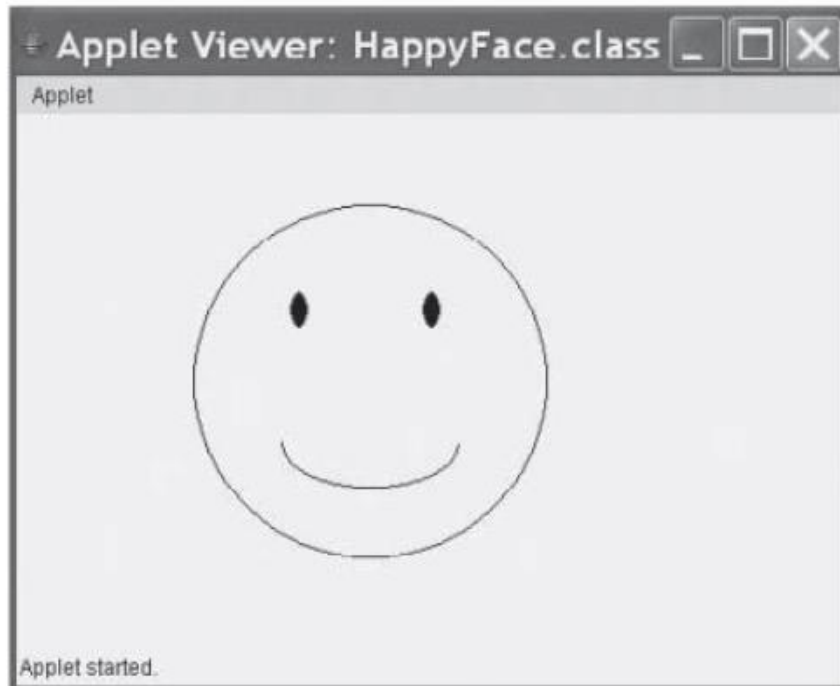


FIGURE 1.6 Screen Coordinate System

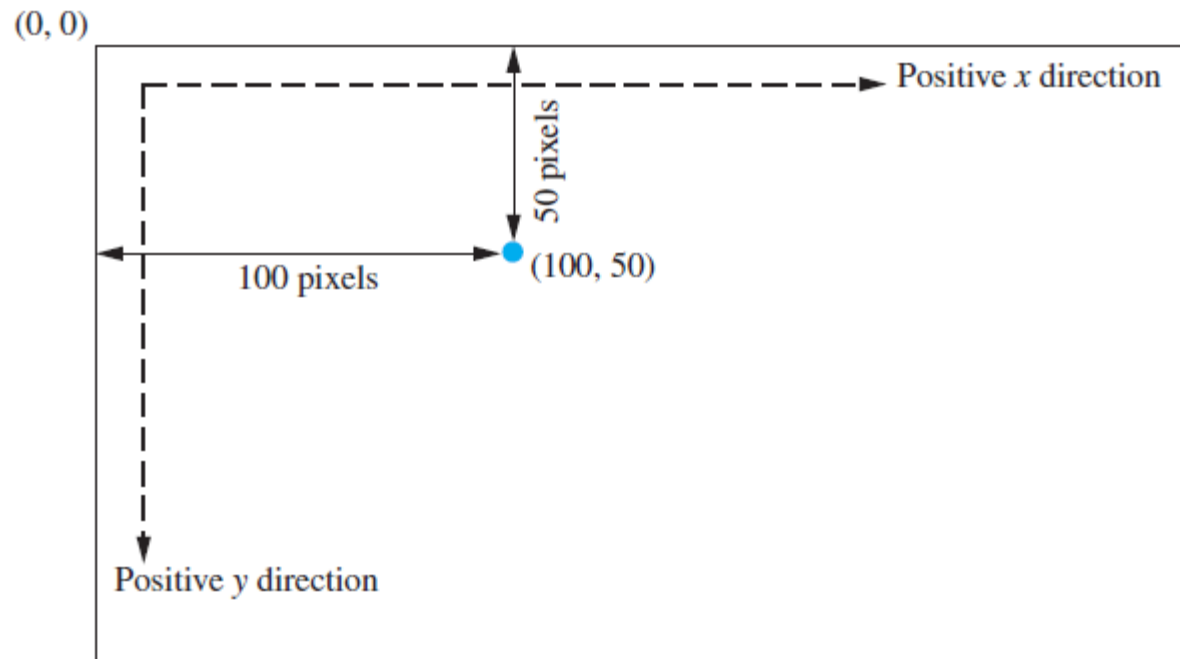


FIGURE 1.7 The Oval Drawn by `canvas.drawOval(100, 50, 90, 50)`

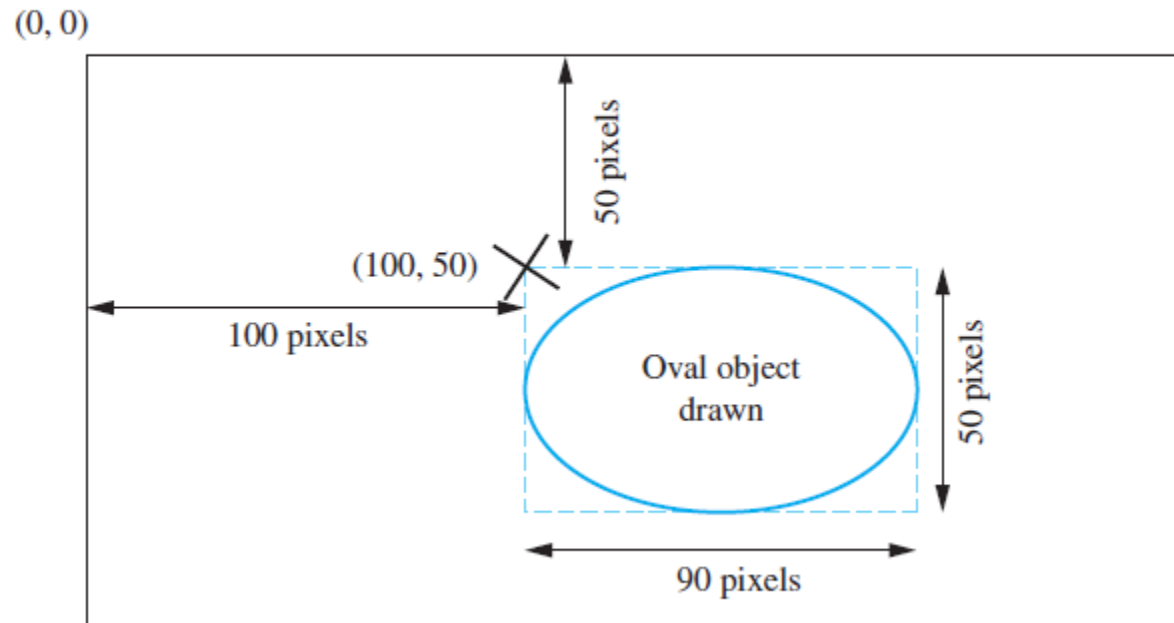
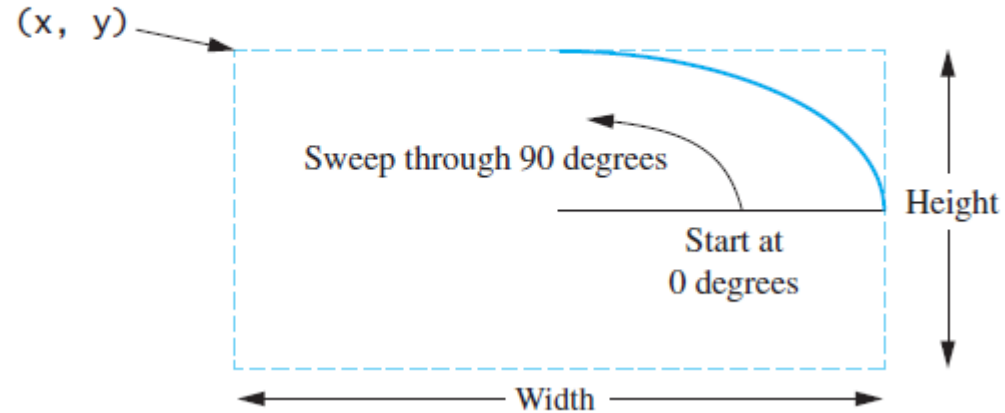
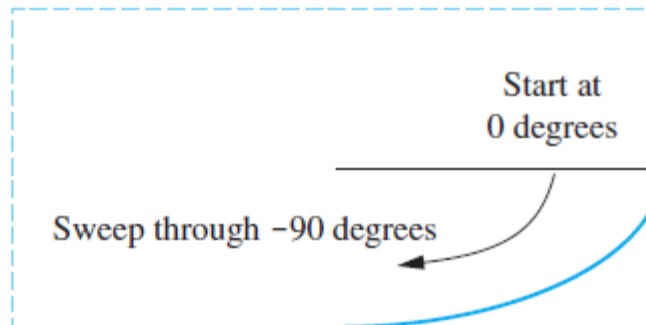


FIGURE 1.8 Specifying an Arc

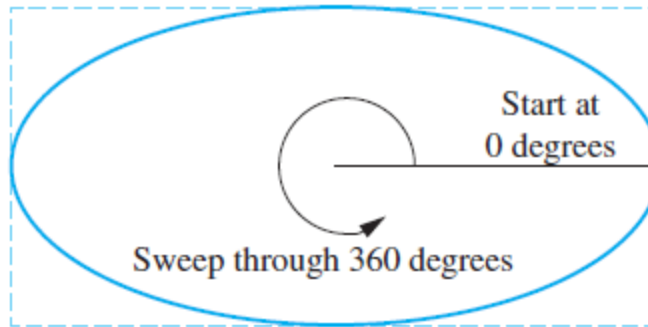
(a) `canvas.drawArc(x, y, width, height, 0, 90);`



(b) `canvas.drawArc(x, y, width, height, 0, -90);`



(c) `canvas.drawArc(x, y, width, height, 0, 360);`



(d) `canvas.drawArc(x, y, width, height, 180, 180);`

