Chocolate bar has grooves that divide it into squares. This is one example. Others would be a bar that is 6 x 12 — big! That’s what I like.

How many cuts are needed to cut the bar of chocolate into all of its pieces? The answer may depend on the initial number of squares and how many grooves there are in each direction.

An instance of class Integer contains, or "wraps", one int value.

You can’t change the value. The object is immutable.

Instance methods: constructors, toString, equals, intValue.

Static components provide extra help.

Each primitive type has a corresponding wrapper class. When you want to treat a primitive value of that type as an object, then just wrap the primitive value in an object of the wrapper class!

<table>
<thead>
<tr>
<th>Primitive type</th>
<th>Wrapper class</th>
<th>Each wrapper class has:</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Integer</td>
<td>• Instance methods, e.g. equals, constructors, toString.</td>
</tr>
<tr>
<td>long</td>
<td>Long</td>
<td>• Useful static constants and methods.</td>
</tr>
<tr>
<td>float</td>
<td>Float</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>Double</td>
<td></td>
</tr>
<tr>
<td>char</td>
<td>Character</td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean</td>
<td></td>
</tr>
</tbody>
</table>

You don’t have to memorize the methods of the wrapper classes. But be aware of them and look them up when necessary. Use Gries/Gries, Section 5.1, and ProgramLive, 5-1 and 5-2, as references.

stepwise refinement

This represents the time of day in a time zone, in terms of hours, minutes, and seconds. The implemented time zones are:

- GMT: Greenwich Mean Time, GMT
- BST: British Summer Time, GMT+1
- EST: Eastern Standard Time, GMT-5 hours (NY)
- EDT: Eastern Daylight Savings Time, GMT-4 hours (NY)
- CST: Central Standard Time, GMT-6 hours (Chicago)
- CDT: Central Daylight Savings Time, GMT-5 hours (Chicago)
- MST: Mountain Standard Time, GMT-7 hours (Phoenix)
- MDT: Mountain Daylight Savings Time, GMT-6 (Phoenix)
- PST: Pacific Standard Time, GMT-8 hours (LA)
- PDT: Pacific Daylight Saving Time, GMT-7 hours (LA)
- IND: India time, GMT+5:30 hours (New Delhi)

India (IND) is included only to show that times are not always on hourly boundaries from GMT.

/ * A time may appear negative or greater than 24 hours. This is because we allow a conversion of a time from one time zone to another, and a time of 0 hours GMT is 24 hours PDT (for example), while a time of 23:59 GMT is 29:29 IND.

An instance of the class can show the time using a 24-hour clock or using the AM-PM designation; it is the user’s choice. */

public class Time {
    public static final String GMT= "GMT";
    public static final String BST= "BST";
    public static final String EST= "EST";
    public static final String EDT= "EDT";
    public static final String CST= "CST";
    public static final String CDT= "CDT";
    public static final String MST= "MST";
    public static final String MDT= "MDT";
    public static final String PST= "PST";
    public static final String PDT= "PDT";
    public static final String IND= "IND";

    private int minutes;  
    private int hours;  
    private int seconds;  
    private int zone;  

    public Time(String time) {  
        // parses the time and sets the zone
    
    public Time(int hours, int minutes, int seconds, int zone) {  
        // initializes the time
    
    public String toString() {  
        // returns the time in the user's zone
    
    public void add(int minutes) {  
        // adds minutes to the time
    
    public void subtract(int minutes) {  
        // subtracts minutes from the time
    
    public void setZone(String zone) {  
        // sets the zone
    
    public String getZone() {  
        // returns the zone

    public static Time convert(int hours, int minutes, int seconds, int zone) {
        // converts a time from one zone to another
    
    public static Time now() {
        // returns the current time

    public static Time test() {
        // tests the Time class

/** Class invariant: Variable time is a time in seconds on a day in time zone zone. The time may be negative or greater than 24 hours, as indicated in class specification (which says why). Field display12Hr has the meaning "the time should be viewed as a 12-hour clock". */

private int time = 0;
private String zone = "GMT";
private boolean display12Hr = false;

/** Constructor: instance with time 0 in GMT and a 24-hour clock */
public TimeJ() {
}

/** Constructor: s seconds, GMT, with 24-hour clock */
public TimeJ(int s) {
    this();
    time = s;
}

/** Constructor: s seconds, zone z, with 12-hour clock iff b is true */
public TimeJ(int s, String z, boolean b) {
    this(s);
    zone = z;
    display12Hr = b;
}

/** Constructor: h hours, m minutes, and s seconds in zone z. The time should be >=24 hours and <=48 hours; if not, 0 is used. If z is not a legal zone, make it GMT. The time should be displayed as am-pm iff b is true */
public TimeJ(int h, int m, int s, String z, boolean b) {
    this(h, m, s);
}

/** = a string representation of the time. This is basically in the form "hours:minutes:seconds zone", but it differs depending on whether a 12- or 24-hour clock is wanted. We describe the difference with examples:

In AM-PM mode, output could be: 06:20:05AM DST
or 06:20:05PM DST

In 24-hour mode: 06:20:05 DST or 18:20:05 DST

If the time is negative or at least 24 hours, print it using the 24-hour mode, even if 12-hour mode is indicated. */

public String toString() {
    int sec; // Field s contains the time in seconds. Local
    int min; // variables hr, min, and sec will contain the corres-
    int hr; // ponding time broken into hours, minutes and seconds.
    String result = ""; // The string to be returned
    boolean amPM; // = "give description in AM-PM format"