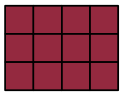
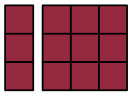
CS1110 Wrapper classes, stepwise refinement 30 September

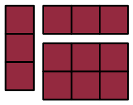
Prelim 7:30-9:00 Thurs, 30 September, Philips 101.



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.



Make a cut by slicing one piece along a groove. To the left, we cut along a vertical groove. Then, we cut one piece along a horizontal groove, shown to the right.



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

CS1110 Wrapper classes, stepwise refinement 30 September

When insults had class

"A modest little person, with much to be modest about." Churchill

"I never killed a man, but I read many obituaries with great pleasure." Clarence Darrow

"Thanks for sending me a copy of your book; I'll waste no time reading it." Moses Hadas

"He can compress the most words into the smallest idea of any man I know." Abraham Lincoln

"I didn't attend the funeral, but I sent a nice letter saying I approved of it." Mark Twain

"I am enclosing two tickets to the first night of my new play. Bring a friend... if you have one." George Bernard Shaw to Winston Churchill

"Cannot possibly attend first night, will attend second... if there is one." Churchill

"I feel so miserable without you; it's almost like having you here." Stephen Bishop

"He is a self-made man and worships his creator." John Bright

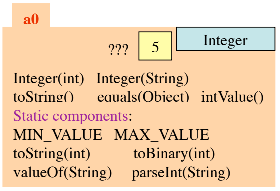
"I've just learned about his illness. Let's hope it's nothing trivial." Irvin Cobb

"There's nothing wrong with you that reincarnation won't cure." Jack Leonard

"He has the attention span of a lightning bolt." Robert Redford

"He inherited some good instincts from his Quaker forebears, but by diligent hard work, he overcame them." James Reston (about Richard Nixon)

Wrapper classes. Read Section 5.1 of class text


a0
???
5
Integer
Integer(int) Integer(String)
toString() equals(Object) intValue()
Static components:
MIN_VALUE MAX_VALUE
toString(int) toBinary(int)
valueOf(String) parseInt(String)

At times, we wish to deal with an **int** value as an object.

"Wrapper class" Integer provides this capability.

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!

Primitive type	Wrapper class
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Each wrapper class has:

- Instance methods, e.g. equals constructors, toString,
- Useful static constants and methods.

```
Integer k= new Integer(63);          int j= k.intValue();
```

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

```
/** An instance 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 -7 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";
```

```
/** 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;
```

7

```
/** 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) {
    }
```

8

```
 /** = 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”
```

9