Arrays

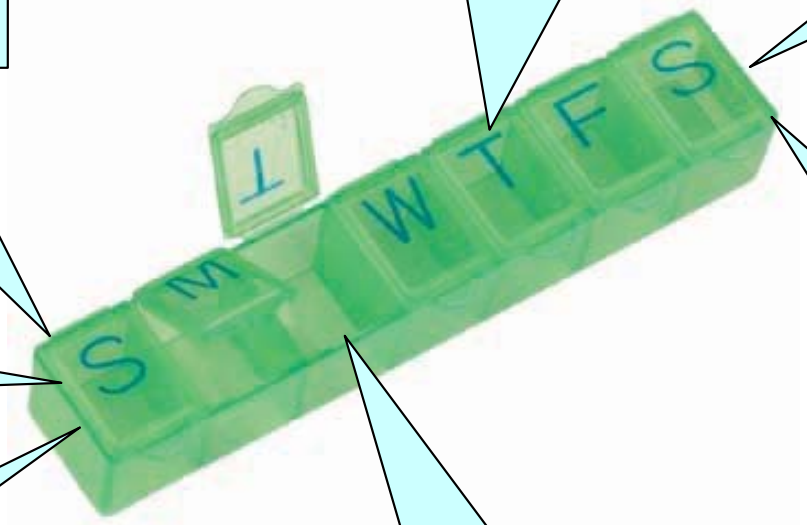
Array concepts

```
Pill[] myPillbox;
```

Variable points to starting point reference

Sequential "memory address"

arr.length == 7



Array element: index 0

fixed size, immutable

1-dimensional

Every element same size, same data type

Simple array coding

- Declaration

```
int[] grades;  // preferred style
int grades[];  // C++ style
```

- Initalization

```
double[] temperatures = new double[10];  // 10 elements
double[] coinVals = {0.01, 0.05, 0.1, 0.25, 1.0, 2.0};
Student[] course = new Student[21]; // any data type
```

- Looping

```
for(int i=0; i < course.length; i++ ) {      // for-loop
for(Student s : course) {                    // for-each-loop
    s.getName();
    s.getStudentId(); ...
```

Array coding uses

- A fixed list of provinces

```
String[] prov = {“ON”,”NS”,”NL”,”NB”,”PE”,”QC”,”MB”,”AB”,”SK”,
   ”BC”,”YT”,”NU”,”NW”};
```

- Match numbers to months

```
String[] months = {“”,“JAN”,”FEB”,”MAR”,”APR”,”MAY”,”JUN”,
   ”JUL”,“AUG”,”SEP”,”OCT”,”NOV”,”DEC”};
month[9] == “SEP”;
```

- Deck of cards

```
String[] suits = {“Spades”, “Clubs”, “Diamonds”, “Hearts”};
String[] ranks = {“A”,”K”,”Q”,”J”,”10”,”9”,”8”,”7”,”6”,
   “5”,”4”,”3”,”2”};
```

Copying arrays

- Strategy 1: for-loop

```
double[] canadaClimate = new double[10];
double[] usClimate = new double[10];
for(int i=0; i < canadaClimate.length; i++)
    usClimate[i] = canadaClimate[i];
```

- Strategy 2: arraycopy

```
double[] canadaClimate = new double[10];
double[] usClimate = new double[10];
System.arraycopy(canadaClimate, 0, usClimate, 0,
  canadaClimate.length);
```

Arrays and methods

- Array as return value

```
public Course[] getTimetable(int studentId) {
  Course[] aList;
  ...
  return aList;
}
```

- Array as argument to method:

```
public void registerForFinalExam(Student[] classroom) {
  ...
```

- Variable number of arguments (*new!*):

```
public void getAvgGrade(double... grades) {
  if(grades.length == 0) {...}// no grades passed
  else if(grades.length > 50) // big class!
     {...}
```

Arrays - Part 2

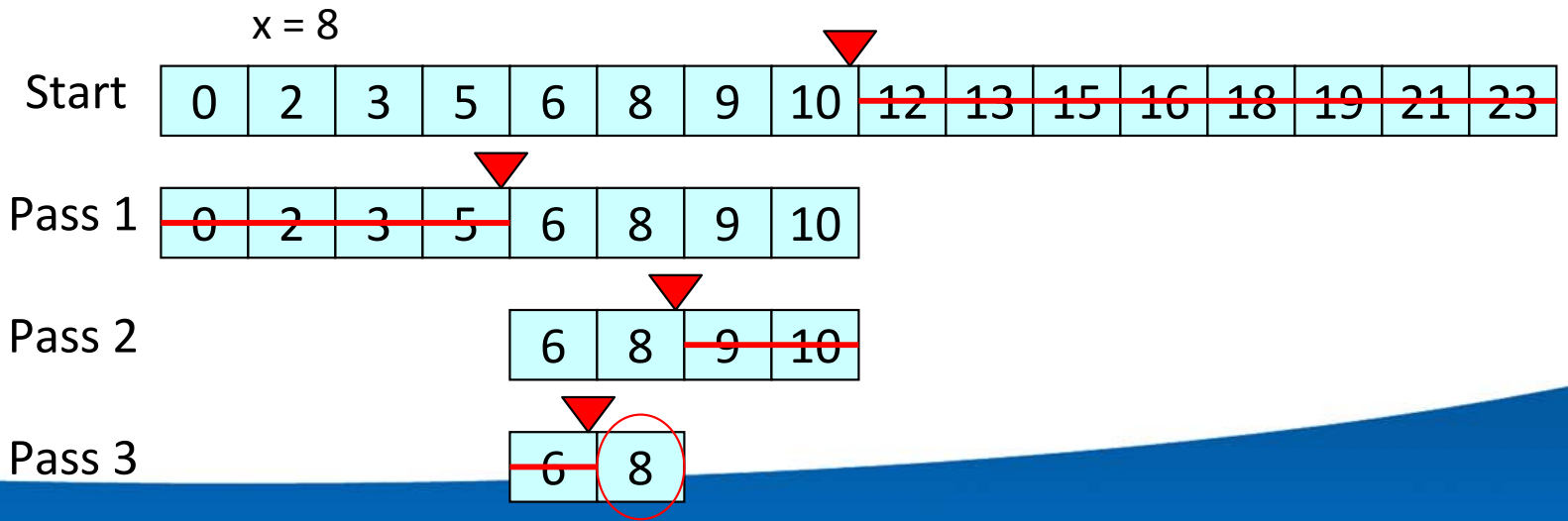
Intermediate arrays

Using arrays for searching

- *Many* different search approaches!
- Linear search is best understood:
  - Search each element, one by one, until element matches **key**
  - Return index when found, or -1 if not (sound familiar?)
  - Pro: elements can be in any order
  - Con: inefficient, especially with large arrays (n/2)

Binary search

- Binary search (p.246):
  - Elements are in ascending order
  - Is key less than, greater than, or equal to key?
  - Pro: more efficient, especially with large arrays ($\log_2 n+1$)
  - Con: elements must be ordered first
- Example:



Sorting arrays

- Lots of sort methods! (Ch6.11)
  - Selection sort
    - Swap smallest for first, swap next for second, etc.
  - Insertion sort
    - Insert an element, compare to the one on the left, adjust
- ...and also
  - Bubble sort, Quick sort, Radix sort, Tree sort, Priority queue sort, Heap sort...

Arrays: the class

- java.util.Arrays
  - Contains convenient static methods for searching and sorting

```
double[] grades;
```

  - Sort:

```
Arrays.sort(grades);
Arrays.sort(grades, start_here, end_here);
```

  - Search:

```
int index = Arrays.binarySearch(grades, 75.0);
```

  - Other cool stuff:

```
Arrays.equals(grades, anotherClassGrades);
Arrays.fill(grades, 75.0); // B's for everybody!
Arrays.toString(grades);
```

Advanced Arrays

Multidimensional arrays

Multidimensional arrays

```
Pill[][] myPillbox =
    new Pill[4][7];
```

`myPillbox[0][0]`

`myPillbox[2][5]`

`myPillbox[3][3]`

myPillbox[row][col]

Sheridan

Simple 2D array coding

- Declaration

```
int[][] grades;  // every element in array is same type
```

- Initalization

```
double[][] temprture = new double[5][3]; // 5 rows, 3 cols
double[][] coinVals = {
  {0.01, 0.05, 0.1, 0.25, 1.0, 2.0},
  {0.01, 0.1, 0.2, 0.5, 1.0},          // 'ragged' array
  {0.05, 0.1, 0.25, 0.5, 1.0}};
```