Arrays

- Arrays...
  - are objects,
  - are dynamically created (via `new`), and
  - may be assigned to variables of type Object.
- An array object contains zero or more *unnamed* variables of the *same* type. These variables are commonly called *elements*.
- A non-negative integer is used to name each element. For example, `arrayOfInts[i]` refers to the $i + 1$*st* element in the `arrayOfInts` array.

Array Types

- An array type is written as the name of an element type followed by one or more empty pairs of square brackets.
  - For example, `int[]` is the type corresponding to a one-dimensional array of integers.
- An array's length is not part of its type.
- The element type of an array may be any type, whether primitive or reference, including interface types and abstract class types.

Array Variables

- Array variables are declared like other variables: a declaration consists of the array's type followed by the array's name. For example,

```
double[][] matrixOfDoubles;
```

  declares a variable whose type is a two-dimensional array of double-precision floating-point numbers.

- Declaring a variable of array type does *not* create an array object. It only creates the variable, which can contain a reference to an array.

- Because an array's length is not part of its type, a single variable of array type may contain references to arrays of different lengths.

Array Variables (cont.)

- To complicate declarations, C/C++-like syntax is also supported, for example,

```
double rowvector[], colvector[], matrix[][];
```

- This declaration is equivalent to

```
double[] rowvector, colvector, matrix[];
```

or

```
double[] rowvector, colvector;
double[][] matrix;
```

- Please use the latter!

Array Creation

- Array objects, like other objects, are created with `new`. For example,

```
String[] arrayOfStrings = new String[10];
```

  declares a variable whose type is an array of strings, and initializes it to hold a reference to an array object with room for ten references to strings.

- Another way to initialize array variables is

```
int[] arrayOf1To5 = { 1, 2, 3, 4, 5 };
String[] arrayOfStrings = { "array",
                            "of",
                            "String" };
Widget[] arrayOfWidgets = { new Widget(),
                            new Widget() };
```

Array Creation (cont.)

- Once an array object is created, it never changes length!

```
int[][] arrayOfArrayOfInt = {{ 1, 2 }, { 3, 4 }};
```

- The array's length is available as a final instance variable `length`. For example,

```
int[] arrayOf1To5 = { 1, 2, 3, 4, 5 };

System.out.println(arrayOf1To5.length);
```

would print "5".

Array Accesses

- All array accesses are checked at run time: An attempt to use an index that is less than zero or greater than or equal to the length of the array causes an IndexOutOfBoundsException to be thrown.

Array Store Exception

- Consider

```
class Point { int x, y; }
class ColoredPoint extends Point { int color; }
class Test {
    public static void main(String[] args) {
        ColoredPoint[] cpa = new ColoredPoint[10];
        Point[] pa = cpa;
        System.out.println(pa[1] == null);
        try {
            pa[0] = new Point();
        } catch (ArrayStoreException e) {
            System.out.println(e);
        }
    }
}
```

Array Store Exception (cont.)

- produces the output:

```
true
java.lang.ArrayStoreException
```