Consider

- Write a program that prompts a user to enter the number of students and then, their names and grades. The program will then outputs the average.

- in addition, the program will prompt the user for a student name, and then return that student’s grade (to double-check the grades entered). This should go on forever until “quit” is entered.
Background

• Programmer often need the ability to represent a group of values as a list
  – List may be one-dimensional or multidimensional

• Array:
  – A Java mechanism for storing an entire list of values
  – of the same type
  – in one location in memory
  – using one variable name
All about arrays

- Array definition
- Array Initialization
- Element access: array subscripting
  - using for loops
- Array as objects
  - data field ‘length’ and methods (equals, clone)
  - array of objects
  - constant arrays
- Copying an array
  - shallow copying
- Array of arrays (multidimensional arrays)
- Array algorithms
  - reverse, search, sort
Array definition

- Without initialization
  `ElementType[] variableName;`

- Examples:
  - `int[] grades;`
  - `String[] studentNames;`
Array definition

- With initialization
  \[
  \text{ElementType[]} \text{ variableName} = \text{new ElementType}[n];
  \]

- Examples:
  - int[] grades = new int[5];
  Or,
  - int numOfStudent = stdin.nextInt();
  - String[] studentNames = new String[numOfStudent];
Array Initialization

- `int[] grades;  // variable `grades` is un-initialized`
- `int[] grades = new int[5];`
  - variable grades is an array of five integers, initialized as 0 by default

- Generally such initializations set the array size and initialize each element as:
  - 0 for integer arrays
  - ‘\0’ for char arrays
  - `false` for boolean arrays
  - `null` for object arrays
Explicit initialization

• Another way to initialize an array is to use a list of:

\[
\text{ElementType[]} \ var = \{\text{exp0, exp1, ... expn}\};
\]

each exp is an expression that evaluates to type ElementType.

Examples:
\[
\text{int[]} \ grades = \{90, 75, 60\};
\]
\[
\text{String[]} \ studentNames = \{"Joe Smith", "Sara Smith"\};
\]
\[
\text{Rectangle[]} \ r = ?
\]
Explicit initialization

- Another way to initialize an array is to use a list of:

```
ElementType[] var = {exp0, exp1, ... expn};
```

each `exp` is an expression that evaluates to type `ElementType`

Examples:
```
int[] grades = {90, 75, 60};
String[] studentNames = {"Joe Smith", "Sara Smith");
Rectangle[] r = {new Rectangle(2,4), new Rectangle(3,1)};
```
Explicit initialization

- Example
  ```java
  String[] puppy = { "nilla", "darby", "galen", "panther" };
  int[] unit = { 1 };
  ```

- Equivalent to
  ```java
  String[] puppy = new String[4];
  puppy[0] = "nilla";
  puppy[1] = "darby";
  puppy[2] = "galen";
  puppy[3] = "panther";

  int[] unit = new int[1];
  unit[0] = 1;
  ```
Element access: array subscripting

- The element of an array variable, say `var`, can be accessed by `var[0]`, `var[1]`, ..., `var[n-1]`, or more precisely,
  \[
  \text{var[subscript-expression]}
  \]

- The `subscript-expression` has to evaluate to integer type with a value that falls between 0 and `n-1`, where `n` is the number of elements

- Example:
  - `int[] grades = {90, 75, 60};`
  - `int total = grades[0] + grades[1] + grades[2];`
Consider

- Segment
  ```java
  int[] b = new int[100];
  b[-1] = 0;
  b[100] = 0;
  ```

- Causes two exceptions to be thrown
  - -1 is not a valid index – too small
  - 100 is not a valid index – too large

- `IndexOutOfBoundsException`
Element Access:

- Consider again the example:
  - `int[] grades = {90, 75, 60};`
  - `int total = grades[0] + grades[1] + grades[2];`

- What if we need to add the grades of 200 students, it will be cumbersome to add up the grade in the above manner
Using for loops with arrays

- Assume grades is an array of 200 elements, storing valid grades for 200 students, the following code calculates the average score

```java
int total = 0;
for (int i = 0; i < grades.length; ++i) {
    total += grades[i];
}
int average = total / grades.length;
```
Using loops for initialization

- Suppose we want to write a program that prompts a user to enter the number of students and their grades ...

```java
System.out.print("Enter the number of students: ");
int numOfStudent = stdin.nextInt();
int[] grades;
for (int i=0; i<numOfStudent; ++i) {
    System.out.print("enter the grade for student "+i+" : ");
    grades[i] = stdin.nextInt();
}
```

Any error with the above code?
Using loops for initialization

- Suppose we want to write a program that prompts a user to enter the number of students and their grades...

```java
System.out.print("Enter the number of students: ");
int numOfStudent = stdin.nextInt();
int[] grades = new int[numOfStudent];
for (int i=0; i<numOfStudent; ++i) {
    System.out.print("enter the grade for student " + i + " : ");
    grades[i] = stdin.nextInt();
}
```

Any error with the above code?
Review

- **Definition**
  - without initialization (e.g., `int[] grades;`)
  - with initialization (e.g., `int[] grades = new int[200];`)

- **Initialization**
  - explicit initialization (`int[] grades = {75, 90};`)
  - using for loops

- **Element access** (syntax: `var[subscript-expression]`)
  - `IndexOutOfBoundsException`