Arrays (2)

Dr. Guang Song
gsong@cs.iastate.edu
Consider

• int age = 20;
  – which defines a primitive type variable ‘age’

• String idiom = “practice makes perfect”
  – which define a reference type variable pointing to a String object

• How about?
  – int[] grades = {90, 65, 75};
  – What kind of variable ‘grades’ is?
Take a look of the memory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>int</td>
<td>20</td>
</tr>
<tr>
<td>idiom</td>
<td>String</td>
<td>0x3468</td>
</tr>
<tr>
<td>grades</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

object String "practice makes perfect"
Take a look of the memory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>int</td>
<td>20</td>
</tr>
<tr>
<td>idiom</td>
<td>String</td>
<td>0x3468</td>
</tr>
<tr>
<td>grades</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

object String “practice makes perfect” 0x3468
Take a look of the memory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>int</td>
<td>20</td>
</tr>
<tr>
<td>idiom</td>
<td>String</td>
<td>0x3468</td>
</tr>
<tr>
<td>grades</td>
<td>Array of int</td>
<td>0x3480</td>
</tr>
</tbody>
</table>

object String “practice makes perfect”

All about arrays

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

- consider
  
  ```java
  int[] grades = {90,65,75};
  int[] newGrades = grades;
  ```

- Is this ok, what is the result of doing this?
Take a look of the memory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>int</td>
<td>20</td>
</tr>
<tr>
<td>idiom</td>
<td>String</td>
<td>0x3468</td>
</tr>
<tr>
<td>grades</td>
<td>Array of int</td>
<td>0x3480</td>
</tr>
<tr>
<td>newGrades</td>
<td>Array of int</td>
<td>0x3480</td>
</tr>
</tbody>
</table>

object String “practice makes perfect”

grades[0] = 80,
grades[1] = 65,
grades[2] = 75
Copying an array (I)

• Suppose we want to make a duplicate copy, so that newGrades has the same elements as grades but is independent, what should we do?
Array members

- Member **length**: Size of the array

  ```java
  for (int i = 0; i < grades.length; ++i) {
    System.out.println(grades[i]);
  }
  ```

- **length** is member variable, not a method. Not to be confused with `length()` (of String class, for example)

- An array knows its own size. This is very different from C.
Array members

• Member clone()
  – Produces a duplicate copy of the values of all the elements in the array

• Now consider:
  ```java
  int[] grades = {90, 65, 75};
  int[] newGrades = grades.clone();
  ```

  what will happen?
Take a look of the memory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>int</td>
<td>20</td>
</tr>
<tr>
<td>idiom</td>
<td>String</td>
<td>0x3468</td>
</tr>
<tr>
<td>grades</td>
<td>Array of int</td>
<td>0x3480</td>
</tr>
<tr>
<td>newGrades</td>
<td>Array of int</td>
<td>0x3504</td>
</tr>
</tbody>
</table>

Now newGrades has its own copy

object String “practice makes perfect”


We can have not only arrays of primitive types, but also arrays of objects. For example, as we have seen in earlier:

- `String[] studentName = {“Joe Smith”, “Sara Smith”};`

which is an array of Strings

Since an array is an object, an array of objects is therefore an object of objects, and therefore there are:

- two levels of referencing
Example

- Rectangle[] r0 = {new Rectangle(2,1), new Rectangle(4,2)};}
Take a look of the memory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td>Array of Rectangles</td>
<td>0x3504</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{r0}[0] &= 0x6600 \\
\text{r0}[1] &= 0x6640 \\
\end{align*}
\]
Now consider

Given: Rectangle[] r0 = {new Rectangle(2,1), new Rectangle(4,2)};

Rectangle[] r1 = r0;
Rectangle[] r2 = r0.clone();

Is r1 or r2 independent of r0?
(i.e., has a separate copy of the two rectangle objects?)
Take a look of the memory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td>Array of Rectangles</td>
<td>0x3504</td>
</tr>
<tr>
<td>r1</td>
<td>Array of Rectangle</td>
<td>0x3504</td>
</tr>
<tr>
<td>r2</td>
<td>Array of Rectangles</td>
<td>0x3520</td>
</tr>
</tbody>
</table>

How to get the two rectangle objects copied?

```
r0[0] = 0x6600
r0[1] = 0x6640
r2[0] = 0x6600
r2[1] = 0x6640
```
Deep copying

Given: Rectangle[] r0 = \{new Rectangle(2,1), new Rectangle(4,2)\};

```java
Rectangle[] r1 = r;
Rectangle[] r2 = r.clone();

Rectangle[] r3 = new Rectangle[r0.length];
for (int i=0; i<r0.length; ++i)
    r3[i] = (Rectangle) r0[i].clone();
```
Take a look of the memory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td>Array of Rectangles</td>
<td>0x3504</td>
</tr>
<tr>
<td>r1</td>
<td>Array of Rectangle</td>
<td>0x3504</td>
</tr>
<tr>
<td>r2</td>
<td>Array of Rectangles</td>
<td>0x3520</td>
</tr>
<tr>
<td>r3</td>
<td>Array of Rectangles</td>
<td>0x3536</td>
</tr>
</tbody>
</table>

- \texttt{r0[0]} = 0x6600
- \texttt{r0[1]} = 0x6640
- \texttt{r2[0]} = 0x6600
- \texttt{r2[1]} = 0x6640
- \texttt{r3[1]} = 0x6680
- \texttt{r3[2]} = 0x6720

- \texttt{object Rectangle(2,1)}
- \texttt{object Rectangle(4,2)}

0x6600 0x6640 0x6680 0x6720
Constant Arrays

• Now consider:
  ```java
  final Rectangle[] r0 = {new Rectangle(2,1), new Rectangle(4,2)};
  ```

  What has been made final, or constant?

• Consider the following statements, which is valid/invalid?
  – r0 = null;
  – r0[0] = null; r0[1] = new Rectangle(4,2);
  – r0[0].resize(4,2);
Given methods zeroInt() and zeroArray(), what are the final values of i and z?

```java
public static void zeroInt(int i) {
    i = 0;
}

public static void zeroArray(int[] list) {
    for (int i=0; i<list.length; ++i)
        list[i] = 0;
    int[] other = {1, 2};
    list = other;
}

int j= 1;
int[] z = {1,2,3};
zeroInt(j);
zeroArray(z);
```
Review

- Arrays are objects
  - member length, methods clone(), equals()
- Arrays of objects
  - two (possible) levels of referencing
- Constant arrays
  - what’s made constant
- Copying an array
  - shallow copying vs. deep copying
- Passing arrays as parameters