Variables & References

• Variables with primitive types:
  – Primitive types
    • Lower case names
    • int, float, double, boolean, …
  – Declare variables:
    • <type name> <variableName>;
    • Memory will be allocated to store one value.
  – Initialization:
    • <type name> <variableName> = <a value>;
    • Without initialization, it might give a default value.
references

• Variable of non-primitive type
  – Syntax of Declarations:
    • <ClassName> <variableName>;
    • ClassName should be in capital
    • Class must be created.
  – Declaration
    • It gives reference
    • It allocates memory for one reference;
    • It will give the initial value null
    • No object (of the class) is created
Examples

• Class Name:
  ```
  class Rational
  {
    int numerator;
    int denominator;
  }
  ```

• Declaration
  ```
  Rational r;
  ```
Object Creation

- Syntax
  - Example

```java
Rational r;
r = new Rational();
```

Heap

<table>
<thead>
<tr>
<th>numerator</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>denominator</td>
<td>0</td>
</tr>
</tbody>
</table>
Object Creation

• properties
  – Use `new` to create
  – Access through reference
  – All objects created on heap

• Assignment
  – No new object is created
  – Assign only the reference.
  – Example
    ```java
    Rational r = new Rational();
    Rational s;
    s = r; // s refers to same object
    ```
initialization

- Use init
  - Example
    
    ```java
    Rational r = new Rational();
    r.init(0,1);
    ```
  - Remember:
    - Line one declares and use default to create an object.
    - Declare w/o object cannot be used to initialize with init.
initialization

• Use the class constructors: example
  
  ```java
  class Rational
  {
    // characteristics
    int numerator;
    int denominator;

    // a constructor
    Rational (int n)
      { numerator = n; denominator=1; }
  } // class end
  ........
  
  Rational r = new Rational(5);
  ```
Array as a reference

- Array variables are references
  - Assign one array-variable to another array variable is copying reference
  - Example:
    ```
    int[] arrA = new int { 5, 7}; // picture in green
    int[] arrB; // picture in light blue
    arrB = arrA; // picture in arrow
    ```

<table>
<thead>
<tr>
<th>arrA</th>
<th>0</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEFE</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

null
Arrays of References

- Array of class type are arrays of references
  ```java
  int[] arrR = new Rational[3]; // picture in green
  arrR[0] = new Rational(4); // picture in light blue
  arrR[1] = new Rational(7); // picture in light blue
  arrR[2] = new Rational(2); // picture in light blue
  ```
Passing parameters

- Passing a reference variable to a method (routine) is passing the reference, not the object.
Reference and memory

- Reference is the address of the allocated memory.
- Memory can only be accessed via reference.
- Memory will NOT be accessible if we don’t have its reference.
- Assign a new reference to a variable does not destroy the object of the old reference.
- Memory of unusable objects are reclaimed by the Java VM.
  - Unusable when there are NO references to it
  - Java VM claims it by its own schedule
  - Reclaim it by activating garbage collection
Garbage Collection

- Algorithm:
  - Go through the stack based references
  - Then via reference from stack trace through the heap and mark the used memory.
  - After the stack is check completely, go back to heap to collect the unmarked memory. Memory will NOT be accessible if we don’t have its reference.

- We can invoke garbage collector
  protected void finalized()
  {
    flush(); flush();
  }