

Comments on Lab 01

Exercise 1.1: Using an Iterator

Initially class E1 looks as follows:

```
package ex01;

import java.util.Hashtable;

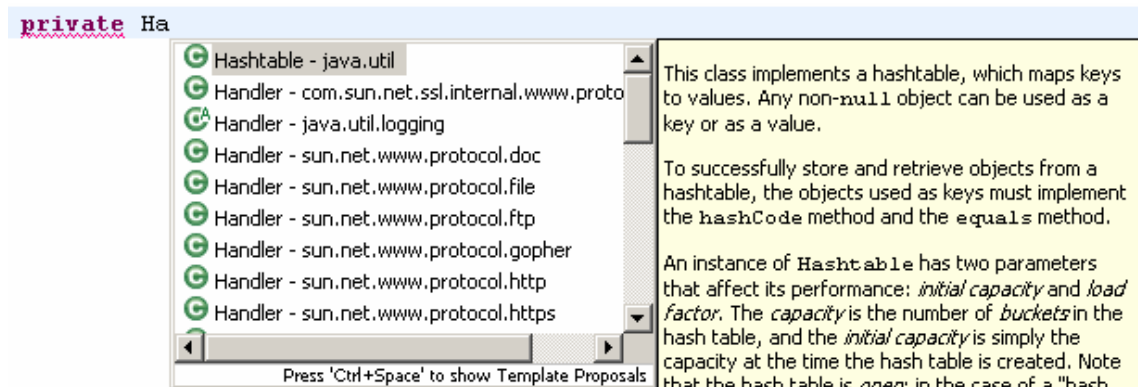
public class E1 {

    private Hashtable<String, String> ht =
        new Hashtable<String, String>();

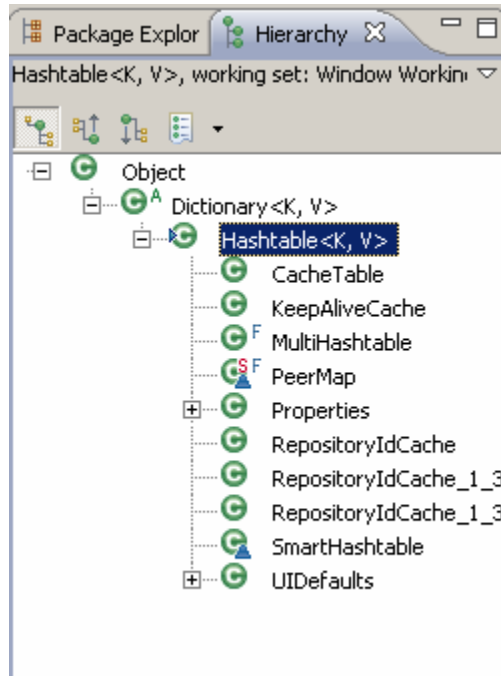
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}
```

As you type the code above, you may start using some time-saving features:

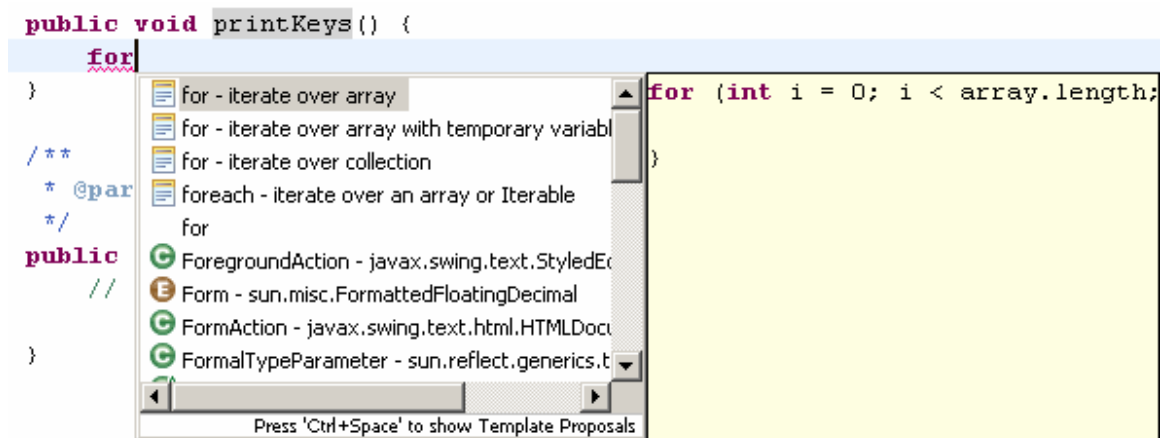
- While being halfway through typing `Hashtable`, for example after typing `Ha`, one may press `Ctrl+Space` to invoke AutoCompletion, which displays in this case a suggestion list as shown below:



- Pressing `F4` while the cursor stands in a type results in the Class Hierarchy for that type being displayed (shown below). Double-clicking on a class displays its source code.



When typing the code for the iteration over the keys of the Hashtable, again we can take a shortcut by using a *template*, a code snippet for a recurring task:



We choose the template labeled “foreach”

The resulting method looks as follows:

```
public void printKeys() {
    for (String k : ht.keySet()) {
        System.out.println("Key : " + k);
    }
}
```

Holding the cursor over a method we don't know (e.g., `keySet()`) results in its Javadoc entry being displayed (this is an invitation for you to document *your* methods with Javadoc, so that later they can be made sense of quickly).

```
htKeys() {  
    k : ht.keySet() {  
    out.println
```

Set<String> java.util.Hashtable.keySet()

keySet

public [Set<K>](#) keySet()

Returns a Set view of the keys contained in this Hashtable. The Set is backed by the Hashtable, so changes to the Hashtable are reflected in the Set, and vice-versa. The Set supports element removal (which removes the corresponding entry from the Hashtable), but not element addition.

Press 'F2' for focus.

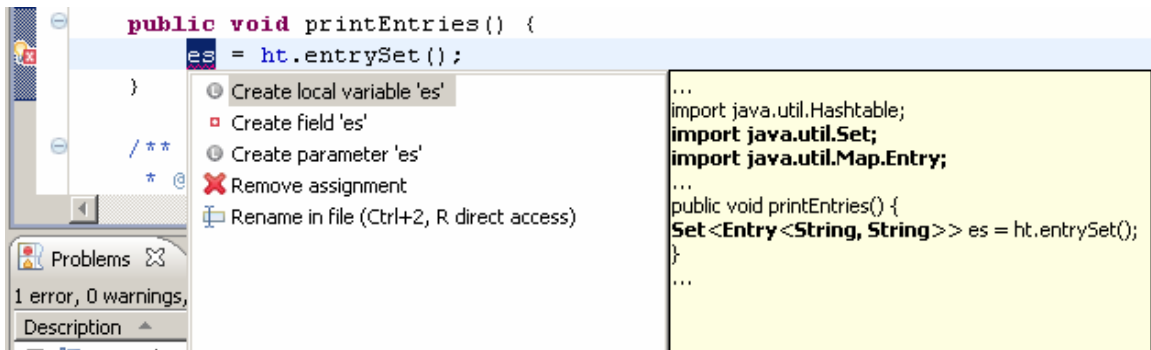
If you like this kind of tutorial, with screenshots illustrating the ideas touched upon in the text, then you might want to know that at <http://www.cs.umanitoba.ca/~eclipse/Eclipse3-1.pdf> the same strategy has been applied to a much wider variety of Eclipse-usage tasks that we can hope to cover in these notes (that PDF is 122 pages long), so you might consider reviewing that text during a programming session with Eclipse to acquire further time-saving skills.

Add a method to print all name value pairs in the form name=value

This can be achieved with the methods we know so far, by retrieving one key at time and then its associated value:

```
public void printEntriesSlightlyInefficiently() {  
    for (String k : ht.keySet()) {  
        String v = ht.get(k);  
        System.out.println("Key : " + k + " , Value : " + v);  
    }  
}
```

However there's a method which achieves the same in just one sweep: `entrySet()`, which returns `Set<Entry<String, String>>`. Notice we need not type the type of that method in full (it's a bit long). Some people prefer to type a variable to hold the result, and let Eclipse come up with the (correct) suggestion for typing that local variable, by clicking on the warning marker on the vertical bar, as shown below:



The option to choose is “Create local variable ‘es’”, after which we get:

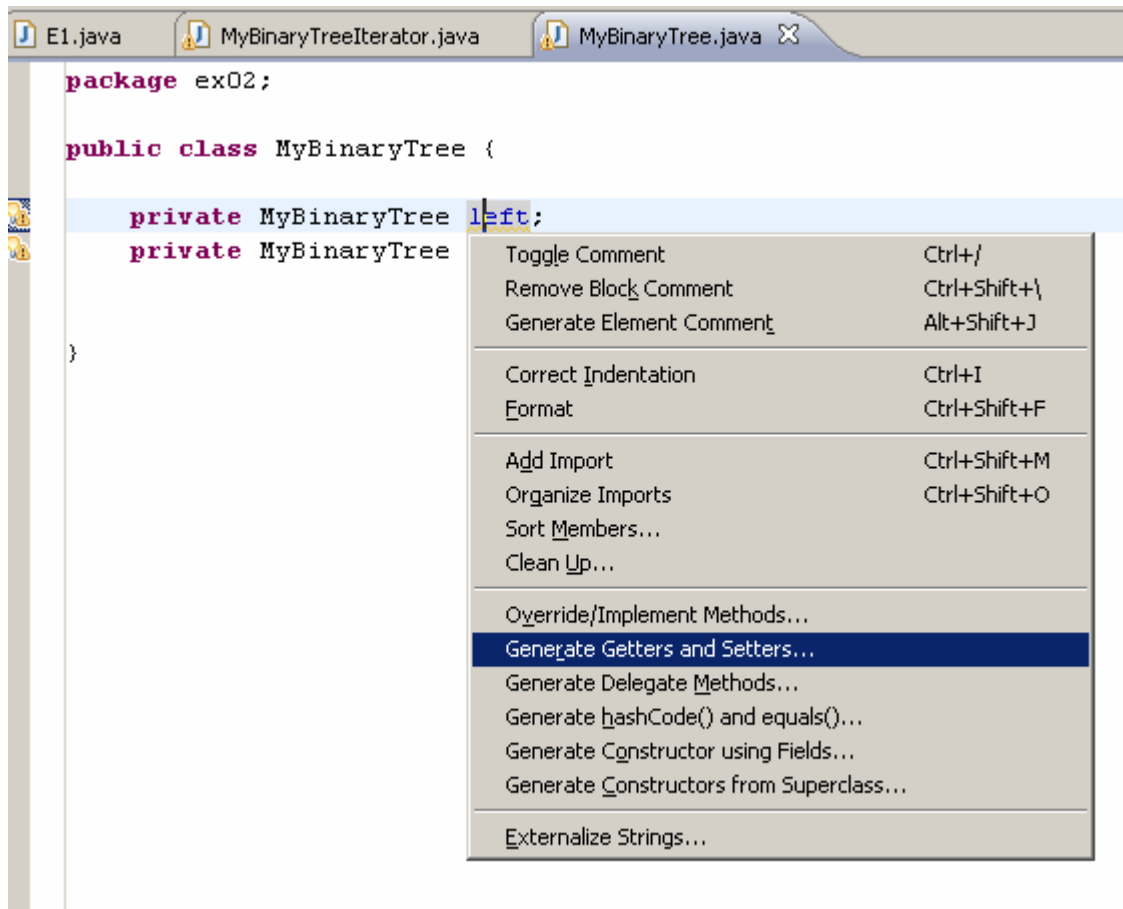
```
public void printEntries() {
    Set<Entry<String, String>> es = ht.entrySet();
}
```

The whole method is:

```
public void printEntries() {
    Set<Entry<String, String>> es = ht.entrySet();
    for (Entry<String, String> entry : es) {
        String k = entry.getKey();
        String v = entry.getValue();
        System.out.println("Key : " + k + " , Value : " + v);
    }
}
```

Exercise 1.2: Implementing an Iterator

While editing `MyBinaryTree`, as with any class with several getters and setters, the following code transformation is useful (just a taste of the more powerful refactorings that Eclipse supports):



The skeleton of the solution to Exercise 1.2 is included in package `ex02`, take a look at method `hasNext()` to derive the logic to complete the other methods (the ones listed with `TODO`).