1. Homework 5

Posted: September/21/2015
Due: September/28/2015 24.00

Use glados.cs.rit.edu, if kansas.cs.rit.edu does not work.

1.1. Homework 5.1 (10 Points)

Given this interface:

```
1  public interface StackInterfaceOld {
2    public void push( Object item );
3    public Object pop();
4    public Object peek();
5    public boolean isEmpty();
6  }
```

Source Code: Src/25/StackInterfaceOld.java

this be a implementation:

```
1/  // this implementation implements the methods,
2/  // but the methods are null methods;
3/  public class StringStackOld implements StackInterfaceOld {
4
5    public void push( Object item ) {}   
6    public Object pop() { return null; } 
7    public Object peek() { return "hi"; } 
8    public boolean isEmpty() { return true; } 
9
10   public static void main(String args[]) {
11       StackInterfaceOld aStackInterfaceOld = new StringStackOld();
12       aStackInterfaceOld.push("hello");    // why is here no warning?
13       String aString = aStackInterfaceOld.pop();
14   }
15  /*
16      javac StringStackOld.java          // explain this error
17      StringStackOld.java:11: incompatible types       // explain what a cast would do
18         String aString = aStackInterfaceOld.pop();
19         // regarding possible compiler err
20         required: java.lang.String
21     */
22
23   }
```

Source Code: Src/25/StringStackOld.java

Explain the questions in CWStackInterfaceOld.java.
See also:
Given this interface:

```java
public interface Storage<E, V> {
    // Appends the specified element to the end of this storage.
    // Returns true, if the element could be added, else false
    boolean add(E e);

    // Inserts the specified element at the specified position in this Storage.
    // Returns true, if the element could be added at position index, else false
    public void add(int index, E element);

    // Adds the specified component to the end of this storage, increasing its size by one.
    public void addElement(E obj);

    // Adds the specified component to the end of this storage, increasing its size by one.
    public void addElement(E obj, V elem);

    // Returns the current capacity of this storage.
    public int capacity();

    // Removes all of the elements from this storage.
    public void clear();

    // Returns a clone of this storage.
    public Object clone();

    // Returns the first component (the item at index 0) of this storage.
    public E firstElement();

    // Returns the element at the specified position in this storage.
    public E get(int index);

    // Returns the last component of the storage.
    public E lastElement();
}
```

Source Code: Src/25/Storage.java

Implement a class using generics for each of the requirements:

- the maximum elements stored is between 0 and 100. You have to use a data structure which capacity is fixed after creation. It is important that adding a item to the storage is O(1). This class has to be named: StorageFixed.java. The compiler can not produce any warning during compilation. You can not use any compiler flags. You can not use any existent java classes for the storage functionally.

- The maximum elements stored is unknown. You have to use a dynamic data structure which capacity is not fixed after creation. This class has to be named: StorageDynamic.java. The compiler can not produce any warning during compilation. You can not use any compiler flags. You can not use any existent java classes for the storage functionally.

You have to provide a test class for your implementation. The same test class has to be used for StorageFixed.java and StorageDynamic.java. It must be possible to compile your implementations with the test class StorageTest.java without any warnings. You can not use any compiler flags.
public class StorageTest {
    public static void main(String args[]) {
        Storage<String, String> aStorageString = new StorageFixed<String, String>();
        Storage<Integer, String> aStorageInteger = new StorageFixed<Integer, String>();
    }
}

Source Code: Src/25/StorageTest.java

% # compilation
% javac StorageTest.java
%

% ssh kansas.cs.rit.edu
# password
# go to the directory where your solution is ...
try hpb-grd lab5-1 "all files you need to compile"

1.2. Homework 5.2 (up to 20 Points)

Given this interface:

public interface Competition<E> {
    // Appends the specified element to this storage.
    // Returns true if the element could be added to this storage
    boolean add(E e);
    // Returns true if this storage contains the specified element.
    boolean contains(E o);
    // Removes the first occurrence of the specified element in this storage.
    // If the storage does not contain the element, it is unchanged.
    boolean remove(E o);
    // Returns the component at the specified index.
    E elementAt(int index);
    // Sorts the storage
    // Returns the sorted storage
    Competition<E> sort();
    // Returns the number of components in this storage.
    int size();
}

Source Code: Src/25/Competition.java

implement a storage class. The compiler can not produce any warning during compilation. You can not use any compiler flags. You can not use any existent java classes for the storage functionally.

The following assumption is true:
• No more than 100000 different string objects will be stored in the storage.
• No more than 1000000 string objects will be stored in the storage.
I will test your program with TestFastCompetition test program:
public class TestFastCompetition {

    int max = 0;
    FastCompetition<String> aStringStorage;

    public TestFastCompetition() {
    }

    public TestFastCompetition(int max) {
        this.max = max;
    }

    private void failure(String reason) {
        System.err.println("You should never see this.");
        System.err.println("Your program did not pass the test");
        System.err.println("Reason: " + reason);
        System.exit(0);
    }

    private void print(String reason) {
        System.err.println("Reason: " + reason);
    }

    private String createString(String theString, int index) {
        return index + theString;
    }

    private void addTest() {
        for (int index = 0; index < max; index ++ ) {
            if ( ! (aStringStorage.add(createString("hello", index) ) )
                failure("add");
            if ( aStringStorage.size() != index+1 )
                failure("size");
        }
    }

    private void addTest2() {
        for (int index = 0; index < max; index ++ ) {
            if ( ! (aStringStorage.add(createString("hello", (int)( max * Math.random())) ) )
                failure("add");
            if ( aStringStorage.size() != index+1 )
                failure("size");
        }
    }

    private void addTest3() {
        for (int index = 0; index < max; index ++ ) {
            if ( ! (aStringStorage.add(createString("hello", 1 ) ) ) )
                failure("add");
            if ( aStringStorage.size() != index+1 )
                failure("size");
        }
    }

    private void containTest() {
        for (int index = 0; index < max; index ++ ) {
        }
    }

    private void containTest2() {
        for (int index = 0; index < max; index ++ ) {
        }
    }

    private void containTest3() {
        for (int index = 0; index < max; index ++ ) {
        }
    }

    private void containTest() {
        for (int index = 0; index < max; index ++ ) {
        }
    }

    private void containTest2() {
        for (int index = 0; index < max; index ++ ) {
        }
    }

    private void containTest3() {
        for (int index = 0; index < max; index ++ ) {
        }
    }
}
if (! (aStringStorage.contains(createString("hello", index)))
    failure("contains");
}

private void containTest3() {
    for (int index = 0; index < max; index++) {
        if (! (aStringStorage.contains(createString("hello", 1)))
            failure("contains");
    }
}

private void sortTest() {
    aStringStorage.sort();

    for (int index = 0; index < max - 1; index++) {
        String thisOne = aStringStorage.elementAt(index);
        String nextOne = aStringStorage.elementAt(index + 1);
        if (thisOne.compareTo(nextOne) > 0)
            System.out.println("thisOne.compareTo(nextOne) >= 0 = " + thisOne + " nextOne = " + nextOne);
    }
}

private void removeTest() {
    for (int index = 0; index < max; index++) {
        if (! (aStringStorage.remove(createString("hello", index)))
            failure("remove " + index);
        if (aStringStorage.size() != max - index - 1)
            failure("remove.size");
    }
    /*
    if (! (aStringStorage.contains("hello" + 1)))
        print("contains - expected");
    if (! (aStringStorage.remove("hello" + 1)))
        print("remove - expected");
    */
}

private void removeTest3() {
    for (int index = 0; index < max; index++) {
        if (! (aStringStorage.remove(createString("hello", 1)))
            failure("remove " + index);
        if (aStringStorage.size() != max - index - 1)
            failure("remove.size");
    }
    /*
    if (! (aStringStorage.contains("hello" + 1)))
        print("contains - expected");
    if (! (aStringStorage.remove("hello" + 1)))
        print("remove - expected");
    */
}

private void stressTest(FastCompetition<String> aStringStorage) {
    this.aStringStorage = aStringStorage;
    addTest();
08       containTest();
09       // sortTest();
10       removeTest();
11   }
12   private void stressTest2( FastCompetition<String> aStringStorage) {
13       this.aStringStorage = aStringStorage;
14       addTest2();
15   }
16   private void stressTest3( FastCompetition<String> aStringStorage) {
17       this.aStringStorage = aStringStorage;
18       addTest3();
19       removeTest3();
20   }
21
22   public static void main(String args[]) {
23       int max = 100000;
24       // int max = 1000;
25       TestFastCompetition aTestFastCompetition = new TestFastCompetition;
26       long startTime = System.currentTimeMillis();
27       FastCompetition<String> aFastCompetition = new FastCompetition;
28       aTestFastCompetition.stressTest(aFastCompetition);
29       aFastCompetition = new FastCompetition<String>(max);
30       aTestFastCompetition.stressTest2(aFastCompetition);
31       aFastCompetition = new FastCompetition<String>(max);
32       aTestFastCompetition.stressTest3(aFastCompetition);
33       long endTime = System.currentTimeMillis();
34       System.out.println(endTime-startTime);
35   }
36
37   }
38
39   Source Code: Src/25/TestFastCompetition.java

You will receive no more than 3 points, if your implementation does not compile with this test program.
You will receive not more than 6 points, if your program fails with this test program.

For hp’s section only:
• The 5 fastest can get up to 10 points,
• The 6-10 fastest can get up to 9 points,
• The 11-15 fastest can get up to 8 points,
• The 16-n can get up to 7 points,

Let’s assume your program is in the group of 6-10, then we will subtract points for mistakes you made.

% ssh kansas.cs.rit.edu
# password
# go to the directory where your solution is ...
try hpb-grd lab5-2 "all files you need to compile"