Foundations of Computer Science
Exam dated 2 July 2025
Programming Test

Stack of Objects

The Java interface below defines the stack abstract data type, that is a container of objects that uses the Last-In-First-Out strategy to insert/extract data items.

```java
public interface Stack {
   /**
      Checks if this stack contains the specified item.
      @param item the specified item to be checked
      @return true if this stack contains the specified item
   */
   boolean contains(Object item);

   /**
      Returns true if this stack is empty
      @return true if this stack is empty.
   */
   boolean isEmpty();

   /**
      Makes this stack empty.
   */
   void makeEmpty();

   /**
      Removes the object at the top of this stack and returns that object as
      the value of this function.
      @return The object at the top of this stack.
      @throws EmptyStackException if this stack is empty
   */
   Object pop() throws EmptyStackException;

   /**
      Pushes an item onto the top of this stack, if not null.
      @param item the item to be pushed onto this stack.
   */
   void push(Object item);

   /**
      Returns the number of item in this stack.
      @return the number of item in this stack
   */
   int size();

   /**
      Looks at the object at the top of this stack without removing it from the
      stack.
      @return the object at the top of this stack.
      @throws EmptyStackException if this stack is empty.
   */
   Object top() throws EmptyStackException;
}
```

where EmptyStackException is the following class:

```java
public class EmptyStackException extends RuntimeException {
   /** Constructs a new EmptyStackException without error message string. */
   public EmptyStackException() { super(); }
   /** Constructs a new EmptyStackException with error message string.
       @param message the error message string.*/
   public EmptyStackException(String message) { super(message); }
}
```

Code first the class **public class S implements Stack** that implements a stack where the constructor constructs an empty stack.
Then code the following extended stack
public class ES extends S where:
- public ES() constructs an empty extended stack
- public ES(ES s) constructs an extended stack that contains the same items as the specified stack and in the same LIFO sequence
- public boolean contains(Object item) overrides the superclass method of the same name using a recursive implementation
- public Object[] toArray returns an array which contains the item of this extended stack in the LIFO sequence (top item is at index 0). The execution of this method does not change this extended stack.

In the code evaluation, the following commands will be used

```
$ javac TestS.java
$ java TestS
$ javac MainES.java
$ java MainES
```

Where the TestS and MainES test classes are below.

```
public class TestS {
   public static void main(String[] args) {
      // define array with test data
      final String[] PEOPLE = {"Eric", "Diana", "Claire", "Bart", "Alice"};

      // define test stack
      S st = new S();        // test of constructor

      // push data into test stack
      for (String s : PEOPLE) st.push(s);  // test of push method

      // print test stack size
      System.out.println("SIZE : " + st.size()); // test of size method

      // print contains
      System.out.print("CONTAINS: ");
      if (st.contains(PEOPLE[0]))             // test of contains method
         System.out.print("Stack contains " + PEOPLE[0]);
      else System.out.print("Stack does not contain " + PEOPLE[0]);
      String notContained = PEOPLE[0] + PEOPLE[1];
      if (st.contains(notContained)) // test of contains method
         System.out.print(" Stack contains " + notContained);
      else System.out.print(" Stack does not contain " + notContained);
      System.out.println();

      // print top and pop
      System.out.print("TOP&POP : ");
      while (!st.isEmpty()) {  // test of isEmpty method
         System.out.print(st.top() + "/" + st.pop() + " ");
      }
      System.out.println();

      // print test stack size
      System.out.println("SIZE : " + st.size());    // test of size method
   }
}
```

```java
public class MainES {
   public static void main(String[] args) {
      // define array
      final String[] PEOPLE = {"Eric", "Diana", "Claire", "Bart", "Alice"};

      // define test extended stack 1
      ES st1 = new ES();

      // push data into stack 1
      for (String s : PEOPLE) st1.push(s);

      // print to array
      System.out.print("TOARRAY1: ");
      Object[] items1 = st1.toArray();  // test of toArray method
      for (Object o : items1) System.out.print(o + " ");
      System.out.println();

      // print stack size
      System.out.println("SIZE1 : " + st1.size());

      // define test extended stack 2
      ES st2 = new ES(st1);  // test ES(ES s) constructor

      // print stack size
      System.out.println("SIZE2 : " + st2.size());

      // print to array
      System.out.print("TOARRAY2: ");
      Object[] items2 = st2.toArray();  // test of toArray method
      for (Object o : items2) System.out.print(o + " ");
      System.out.println();

      // print contains
      System.out.print("CONTAINS: ");
      if (st1.contains(PEOPLE[0]))            // test of contains method
         System.out.print("Stack contains " + PEOPLE[0]);
      else System.out.print("Stack does not contain " + PEOPLE[0]);
      String notContained = PEOPLE[0] + PEOPLE[1];
      if (st1.contains(notContained)) // test of contains method
         System.out.print(" Stack contains " + notContained);
      else System.out.print(" Stack does not contain " + notContained);
      System.out.println();
   }
}
```

The OKS.txt and OKES.txt text files display printouts on standard output when the TestS and MainES classes are run.
The following scores will be assigned in the code evaluation:
 S: contains 6, pop 6, push 6, top 6 (TOT 24),
 ES: ES(ES) 2, contains if recursive 2, toArray 2 (TOT 6)
one point can be awarded for style and one point for asymptotic time complexity of the stack primitives (push and pop) when they are O(1) in the worst case.
Code that reports compile-time errors is usually considered insufficient.
In the submitted code, the student may not
- use instance variables with access specifiers other than private
- add methods other than private
- use classes from the Java Platform API of packages other than java.lang.

Student files must contain a Java comment as the first line, including the student's first and last name, ID, date, and workstation number.

FOUNDATIONS OF COMPUTER SCIENCE

Name____ID._______________________

Date _______________________ Room:______ ADT work station no.___________________

Undersign only one of the following statements:

□ I submit the following files for evaluation:
( )S.java ( ) ES.java ( ) other____

□ I withdraw from the programming test.

Signature____