The exam is over the following sections from Chapters 5 and 6.

- Chapter 5, Sections 5.1. 5.2, 5.3, 5.4, 5.4, 5.6.
- Chapter 6, Sections 6.1, 6.2, 6.3, 6.4.

You should also review all the code examples that we covered in class.

1. The following two lines both generate compiler warnings. What is wrong with them?

```
Stack<String> stack1 = new Stack();
Stack stack2 = new Stack<String>();
```

2. While `String` is a sub-type of `Object` (that is, every `String` object "is a" `Object` object), it is **not** true that `Bag<String>` is a sub-type of `Bag<Object>`. Explain why. (**Hint:** If `A` "is a" `B`, then everything you can do with `B` you can also do with `A`. What can you do with a `Bag<Object>` that you cannot do with a `Bag<String>`?)

3. What is wrong with the following code fragment? How do you fix it?

```
List<String> listOfStrings = new ArrayList<String>();
String s;
for (s : listOfStrings)
   System.out.println(s);
```

4. The following code is supposed to fill `list3` with every possible sum of elements from `list1` and `list2`. So `list3` should end up holding

   `{11, 12, 13, 21, 22, 23, 31, 32, 33, 41, 42, 43, 51, 52, 53}`

   But the code is incorrect. What is wrong? How could you fix it?

```
List<Integer> list1 = Arrays.asList(10, 20, 30, 40, 50);
List<Integer> list2 = Arrays.asList( 1, 2, 3);
List<Integer> list3 = new ArrayList<Integer>();
for (Iterator<Integer> it1 = list1.iterator(); it1.hasNext();  )
   for (Iterator<Integer> it2 = list2.iterator(); it2.hasNext();  )
   {
      int n = it1.next();
      int m = it2.next();
      list3.add( n + m );
   }
for (int i : list3)
   System.out.print(i + " ");
System.out.println();
```

5. (a) Suppose that `it` is an `Iterator` object. Write a small piece of Java code that prints all the objects of `it` to `System.out`.

   (b) Suppose that `it` is an `Iterator` object that returns `Rectangle` objects. Write a small piece of Java code that inserts all the objects of `it` into a newly created `Stack` object.

6. Suppose we perform the following series of stack operations on a single, initially empty stack:

   $push(5), push(3), pop(), push(2), push(8), pop(), pop(), push(9), push(1), pop(), push(7), push(6), pop(), pop(), push(4), pop(), pop()$.

   Draw a picture of the stack at the point where it contains the maximum number of elements (be sure to indicate the top and bottom of the stack). How many of the above operations had been performed at that point?

7. Suppose we implement a stack using a partially-filled array. What is wrong with storing the top-of-stack at location `[0]` and the bottom of the stack at the last used position of the array?

8. If we use a linked list with a head and a tail reference to implement a stack, which is better and why? Having the top-of-stack at the head of the linked list, or having the top-of-stack at the tail of the linked list?

9. Convert the following expression from postfix to infix notation. Use the minimum number of parentheses needed.

```
6  3  2  4  +  -  *
```

10. Convert the following expressions from infix to postfix notation.

```
1 + 2 + 3 + 4
1 + (2 + (3 + 4))
1 + (2 + 3) + 4
2 * 3  * (9 + (3 - 1) + 4) * (5 - 1)
```

11. Here is an incorrect pseudo code for an algorithm which is supposed to determine whether a String of parentheses is balanced: Give an example of an input string that is made up of only the characters '(' and ')', is unbalanced, but for which this algorithm will return true. Explain what is wrong with the algorithm. Can this algorithm ever incorrectly return false when its input string is a balanced string?

```
boolean isBlanced( String input )
{
   declare a character stack
   while ( input has more characters )
   {
      read a character from input
      if ( the character is a '(' )
         push it on the stack
      else if ( the stack is not empty )
         pop a character off the stack
      else
         return false
    }
    return true
}
```

12. Using the two stack algorithm for evaluating fully parenthesized infix expressions, draw the contents of the two stacks just after the '4' token has been read from the following input string and processed by the algorithm.

```
"( ( ( 2 * 3 ) * ( 9 + ( ( 3 - 1 ) + 4 ) ) ) * ( 5 - 1 ) )"
```

13. Only one of the following four class definitions defines a proper abstract class. Which one of the four is it? For the remaining three classes, briefly explain why it does not properly define an abstract class.

```
class A1
{
   abstract void unfinished();
}

abstract class A2
{
   abstract void unfinished(){ }
}

abstract class A3
{
   abstract void unfinished();
}

abstract class A4
{
   private abstract void unfinished();
}
```

14. Suppose that we have classes `A`, `B`, `C` and `D`. Suppose that `B` is a subclass of `A`, that `C` is a subclass of `B`, and `D` is a subclass of `A`. Suppose that we make the following declarations.

```
A a1 = new A();
A a2 = new C();
D d1 = new D();
```

For each part below, explain what, if any, errors would be caused by the statement in that part. Be sure to consider both compile time and run time errors.

(a) `A a3 = new B();`

(b) `B b1 = new A();`

(c) `B b2 = (B) a1;`

(d) `B b3 = (B) a2;`

(e) `B b4 = (B) d1;`

(f) `B b5 = (C)(A)new D();`