Exercise Sheet 12

1. Given is the following class, and beside a translation of method `m` to byte code (some details were left out). The command **`ldc`** pushes a reference to the specified constant on the stack. Answer the questions below with detailed explanations to confirm your answer.

```
class Example2 {
  void m(Object arg) {
    Object local;
    local = "Hello";
    local.concat(" World!");
    arg.concat("Ohh!");
    …
  }
}
```

```
void m(java.lang.Object)
 0:  ldc  #2; //String "Hello"
 2:  astore 2
 3:  aload 2
 4:  ldc  #3; //String " World!"
 6:  invokevirtual String.concat (…)
 10: aload 1
 11: ldc  #5; //String "Ohh!"
 13: invokevirtual String.concat (…)
```

A. Is there a problem with the use of the local variable `local` in the sourcecode?

B. Is there a problem with the use of the local variable `local` in the bytecode verification?

C. Is there a problem with the use of the parameter `arg` in the sourcecode?

D. Is there a problem with the use of the parameter `arg` in the bytecode verification?

2. Given are the following classes and interfaces:

```
interface IFace {
  void m();
}
class Cl1 implements IFace {
   public void m() { System.out.println("Cl1.m"); }
}
class Cl2 implements IFace {
  public void m() { System.out.println("Cl2.m"); }
}
public class Test1 {
  public static void main( String[] args ) {
     xxx(true);
     xxx(false);
  }

  public static void xxx( boolean param ) {
     IFace iface = null;
     if( param ) { iface = new Cl1();}
     else { iface = new Cl2(); }
     iface.m(); }}
```

A. What type will be calculated for the variable `iface` of the method `xxx` during the bytecode verification?

B. When will the method call `iface.m()` be checked: during compilation, bytecode verification, or execution?

C. How would your answer to B change if `IFace` was a class instead of an interface? What if it was an abstract class?

D. If an attacker was to rewrite the method call `iface.m()` in the corresponding bytecode to a nonexistent call, say `iface.funny()`, what effect would this have on the bytecode verification?

E. How would your answer change in D if `IFace` was a class instead of an interface?

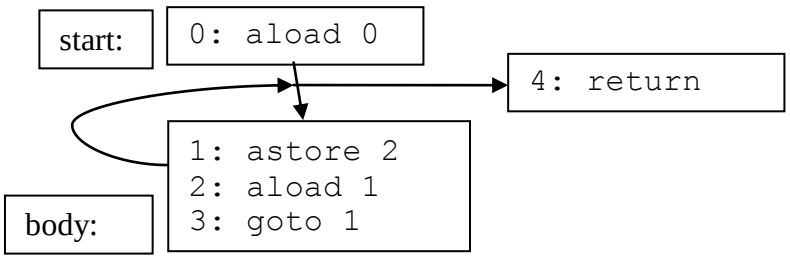
3. For each of the following programs, you must explain the difference between the Java compiler and the bytecode verifier. Assume that the declared class is of type C. For the bytecode examples you must explain the verification step by step. Be detailed!

	Java Program	Bytecode translation

Program 1	`int v = 5;`
	`v = this;`
		`01: iconst 5`
		`02: istore 1`
		`03: aload 0`
		`04: astore 1`

Program 2	`int v = 5;`
	`v = this;`
	`v = v + 1;`
		`01: iconst 5`
		`02: istore 1`
		`03: aload 0`
		`04: astore 1`
		`05: iload 1`
		`06: iconst 1`
		`07: iadd`
		`08: istore 1`

4. Given the example from the lecture:



Where `types(start) = ([],[D,E,T])` and `D <: C` and `E <: C`.

A. Assume that the Java compiler has provided the type assignment `types(body) = ([Object],[C,C,T])`. Verify that the program is still type safe. Give more possibilities for the type assignment for body that would preserve type safety.

B. Assume that instruction 4 has been changed to an aload 0 and then a virtual method call D.m. Which of your type assignments in question A would guarantee type safety?

C. Assume that instruction 4 has been changed to an aload 2 and then a virtual method call C.n. Which of your type assignments would guarantee that this call will be type safe?