

# Homework 2

CSC 4101, Fall 2009

Due: 29 September 2009

1. (10 pts)

Consider the following grammar for variable and class declarations in Java:

```
<Decl>      -> <VarDecl>
             | <ClassDecl>

<VarDecl>   -> <Modifiers> <Type> <VarDec> SEM

<ClassDecl> -> <Modifiers> CLASS ID LBRACE <DeclList> RBRACE

<DeclList>  -> <Decl>
             | <DeclList> <Decl>

<VarDec>    -> ID
             | ID ASSIGN <Exp>
             | <VarDec> COMMA ID
             | <VarDec> COMMA ID ASSIGN <Exp>
```

Indicate any problems in this grammar that prevent it from being parsed by a recursive-descent parser with one token lookahead.

Transform this grammar into a form that can be parsed with a recursive-descent parser with one token lookahead. Make as few changes to the grammar as possible. Do not use Extended BNF.

2. (10 pts)

Consider the following grammar

```
S -> A M
M -> S
   |
A -> a E
   | b A A
E -> a B
   | b A
   |
B -> b E
   | a B B
```

Show a derivation for the string 'a b a a'. Start with

```
S => AM
   => ...
```

and in each step replace one nonterminal by the right-hand side of a grammar rule until you end up with the symbols 'a b a a' on a line by itself. Underline the nonterminal you are expanding and write the expanded line underneath.

3. (10 pts)

Write grammar productions for the Java exception handling constructs `try-catch-finally` and `throw`. The keywords are `try`, `catch`, `finally`, and `throw`. Both the `try-catch-finally` construct and the `throw` construct are statements.

The `try-catch-finally` statement consists of the keyword `try`, a block of statements enclosed in curly braces, zero or more exception handlers, and an optional `finally` block. If there are zero exception handlers, there must be a `finally` block; if there is no `finally` block, there must be at least one exception handler. Each exception handler consists of the keyword `catch`, a single pair of type and parameter name, and a block of statements enclosed in curly braces. A `finally` block consists of the keyword `finally` followed by a block of statements enclosed in curly braces. E.g.,

```
try {
    ...
}
catch (E e) { ... }
catch (F e) { ... }
catch (G e) { ... }
finally     { ... }
```

A `throw` statement consists of the keyword `throw`, an expression, and a semicolon. E.g.,

```
throw new Exception();
```

You can assume that there are nonterminals *Type*, *Expression*, and *Statement*, and a terminal *Ident*. Also, you can use EBNF.

4. (10 pts)

Given the following Java class declarations:

```
class C {
    private int x;

    public int foo() { return this.bar(); }
    public int bar() { return 0; }
}

class D extends C {
    private int y;

    public int bar() { return 1; }
}
```

Show the layout of objects of class D and the contents of class D's virtual function table. What are the values assigned to variables i and j when the following code is executed?

```
C p = new D();
int i = p.bar();
int j = p.foo();
```

Explain which methods get called and why.

5. (10 pts)

Given the following C++ class declarations:

```
class B {
    private:
        int x;
    public:
        virtual int foo() { return this->bar(); }
        virtual int bar() { return 1; }
};
```

```
class C {
    private:
        int x;
    public:
        virtual int foo() { return 2; }
};
```

```
class D : public B, public C {
    private:
        int y;
    public:
        virtual int foo() { return B::foo(); }
};
```

```
class E : public D {
    private:
        int z;
    public:
        virtual int bar() { return 3; }
};
```

Show the layout of objects of class E and the contents of the virtual function tables (you can ignore the `this`-offsets). What are the values assigned to variables `i`, `j`, `k`, and `l` when the following code is executed?

```
B * p = new D();
C * q = new D();
B * r = new E();
C * s = new E();
int i = p->bar();
int j = q->foo();
int k = r->bar();
int l = s->foo();
```

Explain in English which methods get called and why.