CS 164 Programming Language and Compilers **Fall 2015**

Discussion 1: Cool

TAs: Melanie Cebula, Ben Mehne, and Matt Miller

- **Announcement**
  - PA1 due next[, next] Tuesday (9/8) at 11:59pm.
  - WA1/PA2 out next, next Wednesday (9/9).
  - Please contact GSI Ben Mehne to get a class account ASAP if you don't have one.
- **Outline**
  1. Understanding **Cool Syntax, Semantics, and Types**
  2. PA1 QA

Cool

Cool is a Classroom Object-Oriented Language - designed to both be powerful and easy to compile. This discussion will focus on some of the difficult parts of the language - all material discussed here is discussed in more depth in the Cool Manual.

1 Cool Syntax

Which of the following are syntactically valid?

1. `a = b = c`
   Answer:
2. `(* (* nested *) comments are (*in?*)valid*)`
   Answer:
3. ```
   class A {};
   class B { b: Int; };
   class C { bar():Int { 0 }; };
   ```
   Answer:
4. ```
   class Main inherits IO {
       Foobar(a:Int):Int {
           a+1
       };

       main() : SELF_TYPE {
           out_int(0)
       };
   };
   ```
   Answer:

2 Cool Semantics

Cool is designed to be similar to other object-oriented languages and supports inheritance, garbage collection, and static typing.

Cool is also an expression language - most Cool constructs are expressions. Every expression has a value and type. Cool is made up of classes, which are made up of features (attributes/class variables and methods). Each method is made up of expressions (as opposed to statements as in some other languages, like Java).

2.1 Let

1. What is the purpose of let expressions?
   Answer:

2. Can a let expression "redefine" a variable?
   Answer:

3. Is this a valid expression in Cool? If not, why not? If it is valid, what does it evaluate to?
   `let z : Int <- 0, z : Int <- z+1 in z`
   Answer:

2.2 Loops

1. Are loops expressions?
   Answer:

2. What is the type of an evaluated loop?
   Answer:

3. What is the value of a loop after it is completed?
   Answer:

2.3 Blocks

1. What is the value and type of a block expression?
   Answer:

2. Why do blocks exist/why would you need to use them?
   Answer:

2.4 Case

1. How would you implement Cool's `case` in other languages?
   Answer:

2. How else can you control the flow of a program by the dynamic (as opposed to the static) type of an expression?
   Answer:

3. Why might you pick `case` over another method of control evaluation based on dynamic type?
   Answer:

4. What are the runtime errors that case introduces?
   Answer:

3 Cool Types

Cool is a stronlgy typed, object-oriented language with inheritance - it supports the definition of new types in the form of objects and every variable has a type determined at compile time.

1. Is the following program valid? If so, what does it output? If not, why not?

```
class A {
    foobar(value:A):Int {
        1
   };
};

class B inherits A {
    foobar(value:B):Int {
        2
    };
};

class Main inherits IO {
    main() : SELF_TYPE
    {
        let z:A <- new B in out_int(z.foobar(z))
    };
};
```

   Answer:

2. What is the type of the following expression (assuming no runtime error is observed)?

```
case <expr> of
    v : Int => v;
    v : String => v;
esac
```

   Answer:

Further Readings

- Cool Manual (available on the course page)