Introduction to L1

L1 formalizes very basic features of class based oo languages. These are:

- Creation of objects
- State of objects
- Processing Ability of Objects
- Aliasing
- Type System "protects" the integrity of objects at run time

Inheritance will be covered in L2.

Why did we choose these particular features?

Example Execution

We will now look at a small example, which demonstrates the features we will be formalizing in L1.

Consider the following classes:

```
class A{
    int f;
    A next;
    A set(A x){ next = x }
    int incr(){ f = f+1  }
 }
class B{
    int f;
    A next;
    A set(A x){ next = x }
    int incr(){ f = f+20  }
 }
```

The following code demonstrates **unsurprising** aspects of Java.

```
B b1;  A a1; A a2; A a3;

// 1: new C creates new object of class C, for any class C
a1=new A(); a1.f=10; a2=new A; a2.f=20;

//  fields in different objects are (usually) independent
a1.f                                    //
a2.f                                    //

// 2: assignment introduces aliasing
a1 = a2
a1.f                                    //
a2.f                                    //

// 3: methods defined in class are executed by object
a1.incr()
a1.f                                    //

a2.f                                    //
```

```
a3 = new A(); a3.f = 30;

// 4: parameter passing introduces aliasing
a1.set(a3);
a2.next.f                                   //

// 5: aliases are dynamic, i.e. may change
a1 = a3;
a1.f                                        //
a2.f                                        //

// 6: no aliasing for basic types
a1.f = a2.f; a2.f = 40;
a1.f                                        //
a2.f                                        //

 // 7: types
 // b1 = a1;                                 //
```

The answers are:

```
B b1;  A a1; A a2; A a3;
// new C creates new object of class C, for any class C
a1 = new A(); a1.f = 10;
a2 = new A;   a2.f = 20;
//  fields in different objects are (usually) independent
a1.f                                  // 10
a2.f                                  // 20
//   assignment introduces aliasing
a1 = a2
a1.f                                  // 20
a2.f                                  // 20
//   methods defined in class are executed by object
a1.incr()
a1.f                                  // 21
a2.f                                  // 21
```

```
a3 = new A(); a3.f = 30;
//   parameter passing introduces aliasing
a1.set(a3);
a2.next.f                              // 30
//   aliases are dynamic, and may change
a1 = a3;
a1.f                                   // 30
a2.f                                   // 21
//   no aliasing for basic types
a1.f = a2.f; a2.f = 40;
a1.f                                   // 21
a2.f                                   // 40
// 8: type system
// b1 = a1;                                  TYPE ERROR
```

PLEASE PRACTICE IF ANY OF THE ABOVE NOT OBVIOUS