Closure

Function that is created in a particular lexical scope in an environment containing free variables which has access to those free variables after leaving the scope in which it was created

Closure

Function that is created in a particular lexical scope in an environment containing free variables which has access to those free variables after leaving the scope in which it was created

```
int (*)(int) createFunc (int arg) {
   return int (*)(int x) { return arg*x; }
}
```

Closure

Function that is created in a particular lexical scope in an environment containing free variables which has access to those free variables after leaving the scope in which it was created

```
int (*)(int) createFunc (int arg) {
   return int (*)(int x) { return arg*x; }
}

...

int (*)(int) f  = createFunc(3);
printf("%d\n",f(4));
```

Closure

```
List<Integer> squareInts(List<Integer> is){
    List<Integer> result =
       new ArrayList<Integer>(is.size());
    for (Integer i:is) result.add(i*i);
    return result;
}
```

Closure

```
List<Integer> squareInts(List<Integer> is){
    List<Integer> result =
        new ArrayList<Integer>(is.size());
    for (Integer i:is) result.add(i*i);
    return result;
}

is.map({Integer i => i*i})
```

Closure

But this can be simulated using inner classes:

```
class T {
   Thread inner(final int n) {
      return new Thread() {
         public void run() {
            for (int i=0 ; i < n ; i++)
               System.out.print(i+" ");
            System.out.println();
         }
      };
   }
}

public class closure {
   public static void main (String args[]) {
      ((new T()).inner(10)).start();
   }
}
```