COMP 520: Compilers
Problem set 1

Shrey Banga
banga@cs.unc.edu

January 16, 2012

1. ```
   Object o = "abc";
   ```

   (a) ```
       char c = o.charAt(2);
       ```

   This is not a legal Java statement, as the method `charAt` is not defined for the `Object` class to which `o` belongs.

   (b) ```
       boolean b = o.equals("abc");
       ```

   This is a legal statement as the `equals` method is defined for the `Object` class. It returns `true`, which gets assigned to `b`.

   (c) ```
       String s = o;
       ```

   This is not a legal Java statement as a variable of type `String` cannot hold the reference to an instance of the `Object` class, since `String` is a subclass of `Object`. It would have to be explicitly typecast into a `String` object.

   (d) ```
       char d = ((String) o).charAt(3);
       ```

   This is a legal Java statement, as the object is typecast explicitly into a `String` object before calling the `charAt` method on it. However, the index passed to the method is out of the range of the string, causing it to throw a run time exception.

2. The following code shows how to access each of the four occurences of `x`:

   ```java
   interface I1 {int x = 0;}
   class T1 implements I1 {int x = 1;}
   class T2 extends T1 {int x = 2;}
   class T3 extends T1 {
     int x = 3;
     void test() {
       /* Prints 0 */
       System.out.println(I1.x);
   ```

```java
      /* Prints 1 */
      System.out.println(super.x);

      /* Prints 2 */
      System.out.println(new T2().x);

      /* Prints 3 */
      System.out.println(x);
    }
  }

  class Test {
    public static void main(String[] args) {
      new T3().test();
    }
  }
```

3.
```asm
; gcd.asm
; assemble with nasm, link with gcc

global  _main
extern  _printf

section .text
_main:
  ; Save registers we use
  push eax
  push ebx

  mov eax, 84  ; x
  mov ebx, 112 ; y

  ; arguments to printf
  push eax
  push ebx

gcd:
  cmp eax, ebx
  je  print     ; while (x != y) {
  jle else      ;   if(x > y)
  sub eax, ebx  ;     x = x - y
  jmp gcd       ;
else:           ;   else
  sub ebx, eax  ;     y = y - x
  jmp gcd       ;}
```

```nasm
print:
  push  eax ; x and y have already been pushed above
  push  format
  call  _printf
  add   esp, 16

  ; Restore registers
  pop ebx
  pop eax
  ret

format:
  db  'gcd of %d and %d is %d', 10, 0
```