Com S 228
Fall 2014
Exam 1
Sample Solutions

1.

`Skate is = new InlineSkate(3.5, .95);` `System.out.println(is.go(10));`	`0.9 * 3.5 * 0.95 * 10`
`Skate s = new Skate(4.5);` `System.out.println(s.go(10));`	**Compile error:** `Skate is abstract;` `cannot be instantiated` `s = new Skate(4.5);` `    ^`
`Mechanical m = new SkateBoard(.95);` `Skate s = (Skate) m;` `System.out.println(s.go(15));`	**Run-time error:** `ClassCastException`
`LocomotiveDevice ld = new InlineSkate(3.5,` `.95);` `System.out.println(ld.go(25));` `Skate s = (Skate) ld;` `System.out.println(s.getMA());`	`0.9 * 3.5 * 0.95 * 25` `3.5`
`Skate s = new SkateBoard(.95);` `System.out.println(s.go(5));`	**Compile error:** `incompatible types` `Skate s = new SkateBoard(.95);` `          ^` `required: Skate` `found:    SkateBoard`
`Mechanical m = new Bicycle(4.5, .92);` `System.out.println(m.getEfficiency());`	`0.92`

2. a) the try and catch blocks.

```
@Override
public Object clone()
{
        try
        {
                Complex c = (Complex) super.clone();
                return c;
        }

        catch (CloneNotSupportedException e)
        {
                return null;
        }
}
```

b)
```
@Override
public boolean equals(Object o)
{
   if (o == null || o.getClass() ! = getClass())
   {
      return false;
   }

   // typecast o to Complex so that we can compare data members
   ComplexTuple t = (ComplexTuple) o;

   // Compare the data members and return accordingly
   if ((c1 == null && t.c1 == null) &&
       (c2 == null && t.c2 == null))
     return true;
   if ((c1 == null && t.c1 != null) ||
       (c1 != null && t.c1 == null) ||
       (c2 == null && t.c2 != null) ||
       (c2 != null && t.c2 == null))
     return false;
   return c1.equals(t.c1) && c2.equals(t.c2);
}
```

Or

```
@Override
public boolean equals(Object o)
{
   // If the object is compared with itself then return true
   if (o == this)
```

```
        {
           return true;
        }

        /* Check if o is an instance of Complex or not
        * "null instanceof [type]" also returns false */
        if (!(o instanceof ComplexTuple))
        {
           return false;
        }

        // typecast o to Complex so that we can compare data members
        ComplexTuple t = (ComplexTuple) o;
        // Compare the data members and return accordingly
        if ((c1 == null && t.c1 == null) &&
            (c2 == null && t.c2 == null))
          return true;
        if ((c1 == null && t.c1 != null) ||
            (c1 != null && t.c1 == null) ||
            (c2 == null && t.c2 != null) ||
            (c2 != null && t.c2 == null))
          return false;
        return c1.equals(t.c1) && c2.equals(t.c2);
    }
```

3. a) i) Number of iterations of the outer `for` loop: n
   ii) Number of iterations of the inner `for` loop: $n - i$
   iii) Worst-case execution time: $O(n^2)$
   b) i) Number of iterations of the `while` loop: $O(\log n)$
   ii) Time per iteration: $O(n)$
   iii) Total time for the `while` loop: $O(n \log n)$
   iv) Total worst-case execution time for `methodB`: $O(n^2)$
   c) i) Number of recursive calls to `max`: $O(n)$
   ii) Worst-case execution time: $O(n)$
   d) $O(n \log n)$

4. a) Selection Sort.
   b) Insertion Sort.
   c) Quick Sort.
   d) Merge Sort.
   e) Quick Sort.
   f) Merge Sort.