EECS2030 Fall 2018 **Name: (Last, First)** _____
Advanced OOP
Lab Test 3 Exercise Solution
Time Limit: 80 minutes **Student ID** _____

---

1 Programming Exercises

- Download and import this starter project.
- This exercise is based on the Student Management System example discussed in the lectures on inheritance.

2 Written Exercises

These examples questions only cover up to Slide 63 of the inheritance lecture. Similar questions will be covered for later slides of the lecture.

1. Consider the following classes, where we use `print` to abbreviate `System.out.println`:

```
class A extends B {
  A() { }
}
```

```
class B extends C {
  B() { }
}
```

```
class C {
  C() { }
  void bm(){print("C.bm");}
}
```

```
class D extends C {
  D() { }
  void cm(){print("D.cm");}
}
```

```
class F extends D {
  F() { }
  void bm(){print("F.bm");}
  void em(){print("F.em");}
}
```

```
class E extends F {
  E() { }
  void dm(){print("E.dm");}
}
```

Now consider the following code in the `main` method of a tester class for the above classes:

```
1 D d1 = new C();
2 C d2 = new D();
3 d2.bm();
4 D e1 = new E();
5 d2 = e1;
6 d2.bm();
7 F f = e1;
8 e1.em();
```

(a) Explain if **Line 1** compiles.

> **Solution:** No, because the new dynamic type `C` is not a descendant class of `d1`'s static type `D`.

(b) Explain if **Line 2** compiles.

> **Solution:** Yes, because the new dynamic type `D` is a descendant class of `d2`'s static type `C`.

(c) Explain if **Line 3** compiles. If yes, write down and explain how the output is printed.

> **Solution:**
> Yes, because method `bm` is defined in `d2`'s static type `C`.
> The dynamic type of `d2` is `D`, which means the version of method `bm` inherited from class `C` is called: output is `C.bm`.

(d) Explain if **Line 5** compiles. If yes, what are the static type and dynamic type of `d2` after `Line 5` is executed?

> **Solution:**
> Yes, because the `e1`'s static type `D` is a descendant class of `d2`'s static type `C`.
> The static type of `d2` remains as `C`.
> The dynamic type of `d2` changes to `E` (i.e., the dynamic type of `e1`).

(e) Explain if **Line 6** compiles. If yes, write down and explain the output.

Solution:
Yes, because method `bm` is defined in `d2`'s static type `C`.
The dynamic type of `d2` is `E`, which means the redefined/overridden version of method `bm` inherited from class `F` is called: output is `F.bm`.

(f) Explain if **Line 7** compiles.

Solution: No, because `e1`'s static type `D` is not a descendant class of `f`'s static type `F`.

(g) Explain if **Line 8** compiles. If yes, write down and explain the output. If no, suggest a fix using type casting, then write down and explain how the output is printed.

Solution:
No, because `e1`'s static type `D` does not have the method `em` defined.
We can use a type cast: `((E) e1).em()`, which performs a downward cast on `e1` and creates a new reference of static type E, so we can call the method `em`.
Since e1's dynamic type is `E`, so the version of the method `em` in class E, which is inherited from class `F`, will be called: output is `F.em`.