Advanced Object Oriented Programming and Design

Theoretical Homework 1

Abraham Murciano and Daniel Klein

October 27, 2020

Part 1: Interface Definition

We are given the following code which we can assume is correct.

```
public static void main(String[] args) {
    int x;
    String y;
    C b1 = new B();
    A[] arrA = new A[3];
    arrA[0] = new B(x, y);
    arrA[1] = new B();
    arrA[2] = new C();
    arrA[0].f();
    arrA[0].g();
    arrA[0].h();
}
```

Section A

We are told A is an interface, and we are tasked with writing its declaration.

```
interface A {
    void f();
    void g();
    void h();
}
```

Section B

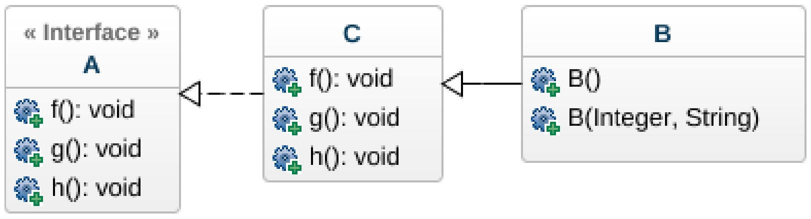
We must now implement classes B and C.

```
class C implements A {
    void f() {}
    void g() {}
    void h() {}
}

class B extends C {
    public B() {}
    public B(int, String) {}
}
```

Section C

Below is a UML diagram for the classes A, B, and C.



Section D

C cannot be an interface, because we call its constructor in the code, and interfaces cannot be instantiated.

Part 2: Plane and Train System

If our plane and train classes would implement the `Comparable` interface, then when calling `sortTarnsport` on an array containing both of these different types, we would get an error saying that we cannot cast from one of the types to the other. However, if these classes implement the generic `Comparable<T>` interface, where `T` is the class `Train` or `Plane`, then our code would not allow for such a circumstance to occur, since we must pass it specifically an array of `Train` objects or `Plane` objects, not anything else.