THE UNIVERSITY OF BRITISH COLUMBIA
CPSC 210: MIDTERM EXAMINATION – February 7, 2017

Last Name:	Zhao	First Name:	Steven
Signature:	[illegible]	UBC Student #:	1 8 9 4 9 1 6 4

Important notes about this examination

1. You have **60 minutes** to write this exam.
2. **You should have received a separate collection of pages including Java code.**
3. **All questions must be answered on this portion of the exam.**
4. **You may complete the exam in either pen or pencil.**
5. Put away books, papers, laptops, cell phones… everything but pens, pencils, erasers and this exam.
6. Good luck!

Please do not write in this space:

Question 1a,b:	8	HM
Question 1c,d:	9	TK
Question 2:	6	[illegible]
Question 3:	15	TA
Question 4a:	7	[illegible]
Question 4b:	7	JL
Question 4c:	4	ST

0 236366 453321

00257

Use this page if you need more space to answer a question.

Note: All the code you need to refer to in writing this exam appears in the code booklet. It has been slightly modified by removing some `package` and `import` statements.

1. EXTENDING EXISTING CLASSES AND WRITING TESTS [20 points]
Consider the `BooleanCollection` class, which is identical to what you saw in Assignment 3 and appears on the first page of the code section of the exam.

2 **(1.a) [4 points]** Write a *specification* for the following method to be added to the `BooleanCollection` class:

The method `removeThese` takes a single parameter, a boolean `b`, and removes from the collection all those elements that are equal to `b`. So, if `b` is true then `removeThese` removes all the true elements and if `b` is false then `removeThese` removes all the false elements.

```
// MODIFIES: this ✓
// EFFECTS: removes all true in collection if b is equal to
   true, remove all false otherwise ✓
```

Method Stub?

6 **(1.b) [6 points]** We would like to add a `sameUpTo` operation to our `BooleanCollection` class which returns true if this `BooleanCollection` and one passed as an argument have the same elements in the same order up to but not including the element at the provided `index`. The specification for the `sameUpTo` method is:

```
// REQUIRES: 0 <= index <= smallest of: getNumberOfItems() and
// other.getNumberOfItems()
// EFFECTS: return true if this and other have the same elements in
// the same order, up to but not including the element at the
// provided index, false otherwise
public boolean sameUpTo(BooleanCollection other, int index) {
    for (int i = 0; i < index; i++) ✓ {
        if(this.get(i) != other.get(i)) {
            return false; ✓
        }
    }
    return true; ✓
}
```

00257

(1.c) [4 points] Write a single test method that tests the `BooleanCollection`'s `sameUpTo` method when both `this` and `other` are empty and the argument `index` has the value 0. This method would be in, for example, a `BooleanCollectionTest` class. Assume that no `@Before` method exists - if you wish to make use of one, you must include it in your answer.

```
@Test
public static void test(){
      BooleanCollection testBooleanCollection = new BooleanCollection();
      BooleanCollection otherCollection = new BooleanCollection();

      assertTrue(testBooleanCollection.sameUpTo(otherCollection,0));

}
```

4

(1.d) [6 points] Write a single test method that tests the `BooleanCollection`'s `sameUpTo` method when `this` and `other` have the same first 2 elements in the same order, but `other` has one extra element. Your test should check that `sameUpTo` produces the correct result when asked to determine if the first 2 elements in the collections are the same and in the same order. As in 1.c above, this method would be in, for example, a `BooleanCollectionTest` class. Assume that no `@Before` method exists - if you wish to make use of one, you must include it in your answer.

```
@ Test
public Static void test2(){
      BooleanCollection testBooleanCollection = new BooleanCollection();
      BooleanCollection otherCollection = new BooleanCollection();

      testBooleanCollection.add (true);
      testBooleanCollection.add (false);
      otherBooleanCollection.add(true);
      otherBooleanCollection.add(false)
      otherBooleanCollection.add (true);

      assertTrue(testBooleanCollection.sameUpto(otherCollection,3));

}
```

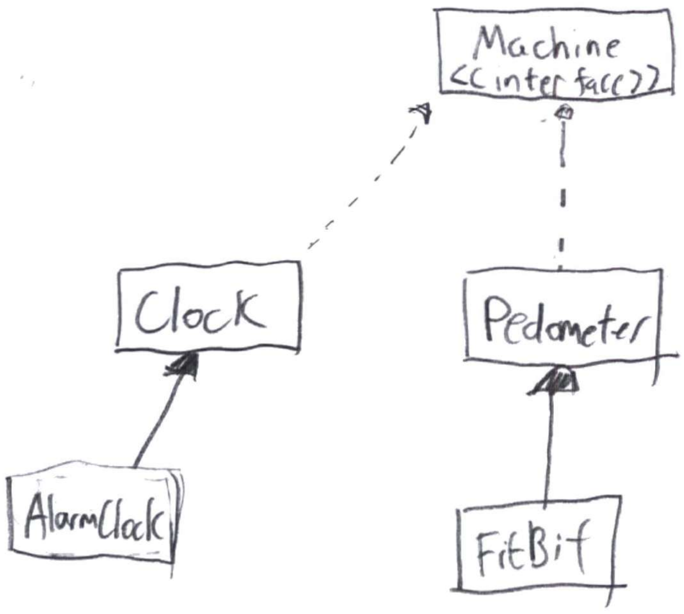
5

(-1) should call with 2

00257

2. TYPE HIERARCHIES [6 points]

Draw the type hierarchy that includes all the types in the `model` package of the `Machines` project (the code is attached – note that it spans several pages). Just include type names and extends/implements relationships between types. Do NOT include methods or fields (the diagram will just get too big).



3. APPARENT AND ACTUAL TYPES [16 points]

Assume the following variables have been created with the indicated types and have been correctly initialized. The initial value of these variables does not affect this question in any way. None of these variables has the value `null`.

```
Machine machine;
Clock clock;
AlarmClock alarmClock;
Pedometer pedometer;
FitBit fitBit;
```

(3.a) [12 points] Circle Yes or No indicating whether each of the following sets of statements will compile. For each one explain very briefly (do not exceed the provided space) why it will or will not compile.

`machine = new Machine();`

Will it compile? Yes [✓ No]

2 *Explain:* it is an interface, it's abstract and has no constructor & cannot be initialized

`machine = new Pedometer();`

Will it compile? [✓ Yes] No

2 *Explain:* ~~It~~ Pedometer is a class which implements a Machine, apparant type is Machine

`machine = clock;`

Will it compile? [✓ Yes] ~~No~~

2 *Explain:* As Machine is an interface of clock, it can be set.

`clock = machine;`

Will it compile? Yes [✓ No]

1 *Explain:* Clock cannot be set to ~~a~~ its implementation. Machine could be a fitbit for on example.

`pedometer.getGoal();`

Will it compile? Yes [✓ No]

2 *Explain:* Pedometer is a superclass of FitBit and does not have the getGoal method,

```
pedometer = new FitBit();
pedometer.getDistance();
```

Will these statements compile? [✓ Yes] No

Explain:

2 Pedometer has the method getDistance

6

00257

(3.b) [4 points] Suppose there is a method with the following signature (note that the implementation is not important in the context of this question):

```
public void print(Pedometer p) {
    // stub
}
```

(3.b.i) [2 points] Will the following call to this method compile assuming it is made from within the class in which it is declared?

```
print(new Clock());
```

Will it compile? *Yes* [✓ *No*]

2 Explain: Print takes in a pedometer, not a clock

(3.b.ii) [2 points] Will the following call to this method compile assuming it is made from within the class in which it is declared?

```
print(new FitBit());
```

Will it compile? [✓ *Yes*] *No*

Explain:

~~[illegible] yes~~

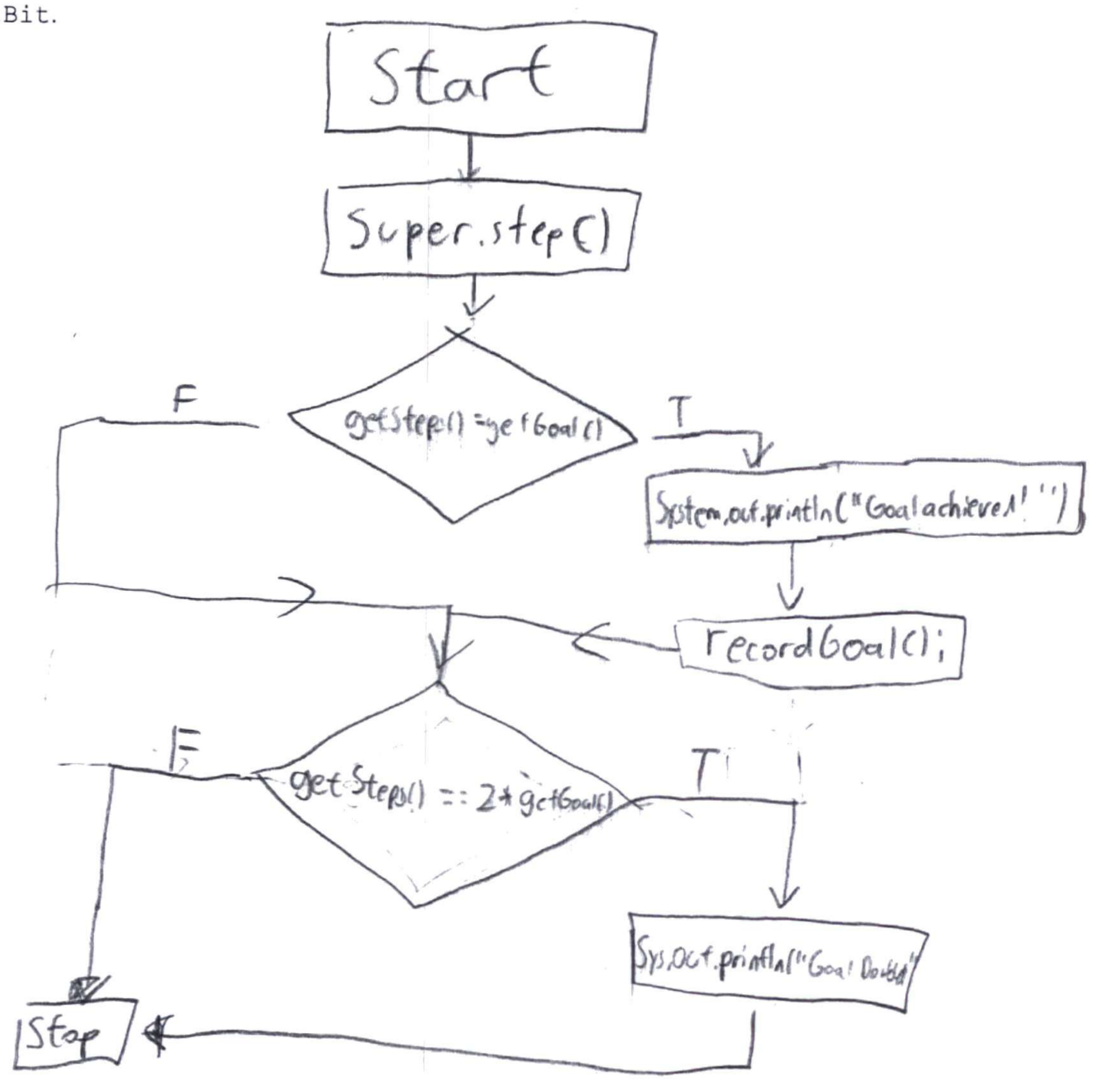
2 ~~It's [illegible] type is not equal to the required argument Pedometer.~~

Yes it does compile as FitBit extends Pedometer and is a subclass of Pedometer

00257

4. CONTROL FLOW [18 points]

(4.a) [7 points] Draw an intra-method flowchart for the method `step` in the Class `FitBit`.

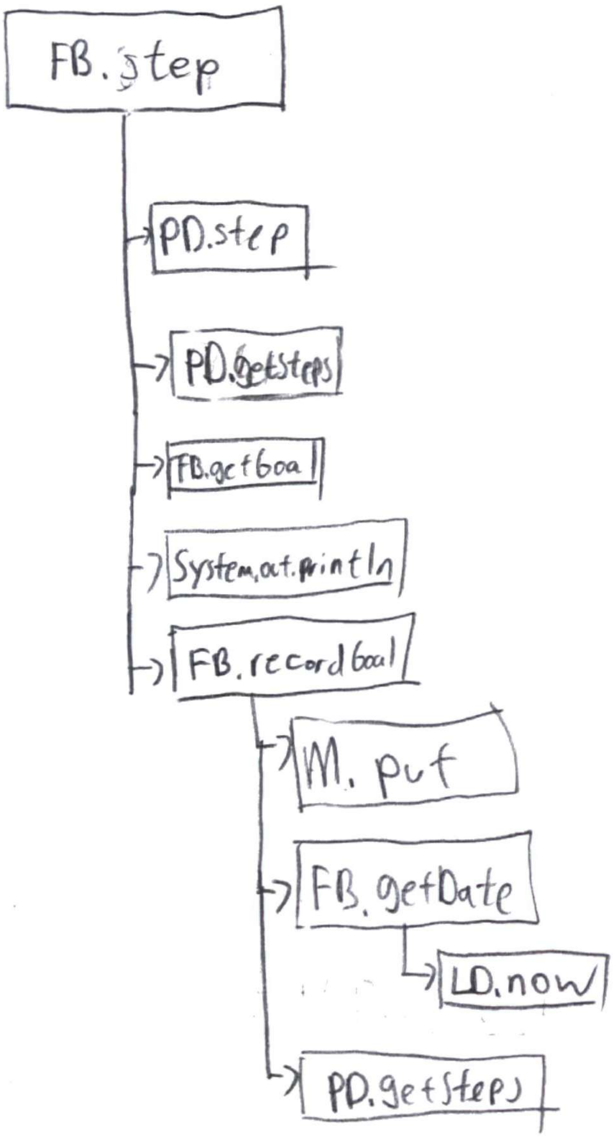

Start
Super.step()
getSteps() == getGoal()
F
T
System.out.println("Goal achieved!")
recordGoal();
getSteps() == 2 * getGoal()
F
T
Sys.out.println("Goal Doubled")
Stop

1

00257

(4.b) [7 points] Draw an inter-method call graph for the method `step` in the Class `FitBit`. Include the first level of calls to the Java library. Be sure to include a legend if you abbreviate type names.

FB = FitBit
PD = Pedometer
M = Map
LD = LocalDate



✓

7

00257

(4.c) [4 points] What is printed out when the following method runs:

```
public static void main(String[] args) {
   List<Pedometer> pedometers = new ArrayList<>();
   Pedometer p = new Pedometer();
   FitBit f = new FitBit();

   pedometers.add(p);
   pedometers.add(f);
   pedometers.add(p);

   for (Pedometer p : pedometers) {
      System.out.println(p.getName());
   }
}
```

Pedometer
FitBit
Pedometer

4