Name:	Number:

Object Oriented Programming 2009/10

Final Exam
July 1st, 2010

Directions (read carefully):
- CLEARLY print your name and ID on every page.
- The exam contains 8 pages divided into 4 parts. Make sure you have a complete exam.
- The point value of each problem is indicated next to the problem and below.
- You have two hours.
- It is wise to skim all the problems before beginning, to best plan your time.
- This is a closed book exam. No notes of any kind are allowed. Do all work in the space provided, using the backs of sheets if necessary.
- **Over the table it should ONLY be this exam, a pen and an ID.**
- Turn off the mobile phone. The use of a mobile phone annuls the exam.

Part	Problem	Description	Page	Marks
I	1 a) b)	UML	2	1.5
II	2 a) b) c) d)	Java development	4	3.0
	3 a) b) c) d) e)	Java multiple choice	5	2.5
	4	Java miscellaneous	7	1.0
III	5	XML	8	1.0
IV	6	C++	8	1.0
Total				**10.0**

Name:	Number:

Part I -- UML (1.5 marks)

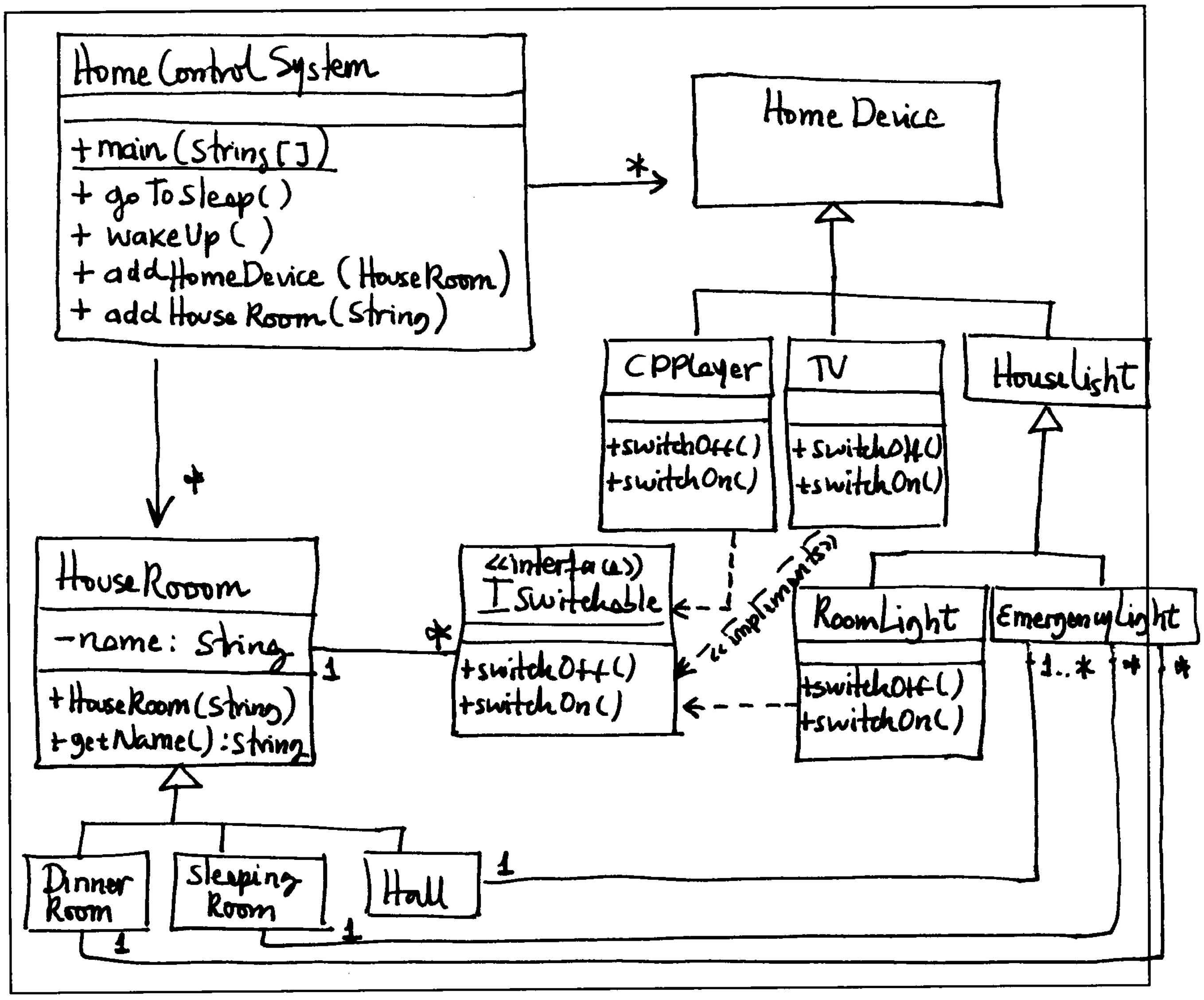
1 – Consider a home system that controls several home devices.

Home devices include CD players, TVs and house lights. House lights include emergency lights and room lights. Room lights, CD players and TV should implement the interface `ISwitchable`, with `switchOff` and `switchOn` methods (without parameters and that return void), which means that they can be automatically powered down by the Home Control System. By contrast, emergency lights are never powered down at night.

Consider that a home device is always associated with one house room. On the other hand, house rooms should have: (i) several home devices; and (ii) a name. Consider that house rooms could only be hallways, dinner rooms and sleeping rooms. The hallway is a house room that always has emergency light devices, whereas dinner and sleeping rooms could have any home device.

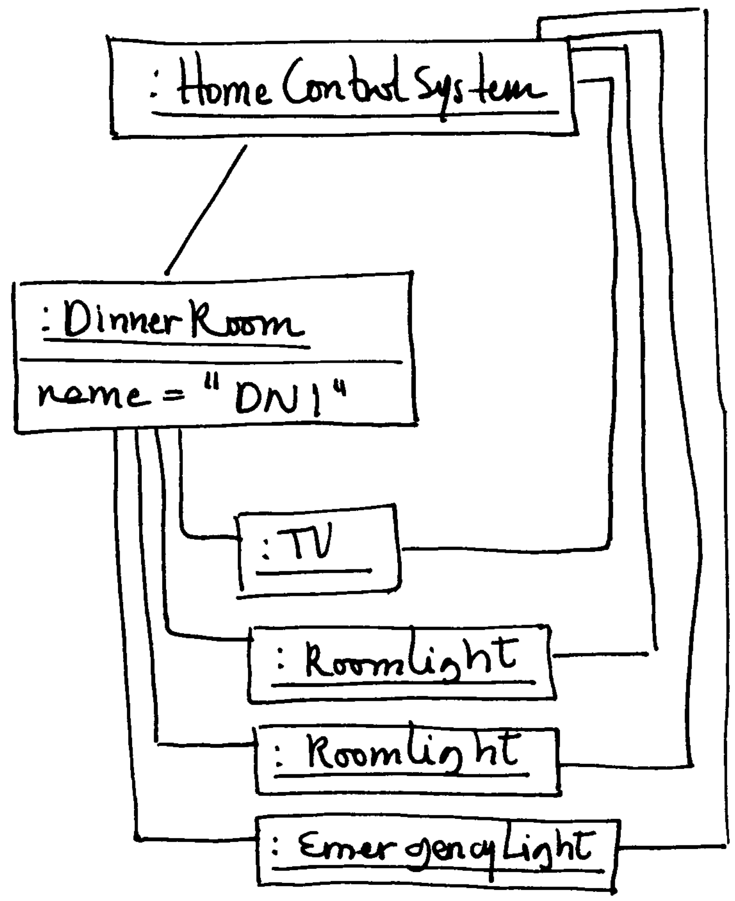
The Home Control System should have a `main` method (as in Java) and four supplementary methods: (i) `goToSleep`, no parameters; (ii) `wakeUp`, no parameters; (iii) `addHomeDevice`, which adds a home device to the Home Control System; and (iv) `addHouseRoom`, which adds a house room to the Home Control System. Home devices exist in the Home Control System always associated with a room house. House rooms are always built with a name.

a) **[1.0 mark]** Define the UML class diagram for the presented problem.



Name: Number:

b) [0.5 marks] Define an UML object diagram considering that a Home Control System has four home devices: one TV, two room lights, and one emergency light. All these home devices are in a same dinner room. Set all needed attributes with some dummy values.



Name: Number:

Parte II -- Java (6.5 marks)

2 – An abstract function is a mathematical entity that takes an input, performs some calculation and produces an output. Such a definition is called a *lambda*. In OOP, we can model abstract functions as an interface, called `ILambda`, that defines a method `apply` that takes an `Object x` as input and returns an `Object` as output.

a) **[0.5 marks]** Define the `ILambda` interface.

ILambda:
```
public interface ILambda {
      public Object apply(Object x);
}
```

b) **[0.5 marks]** Define two concrete *lambda* functions, called `Lambda1` and `Lambda2`, for which the `apply` method should return:

- a `String` that concatenates "`Lambda1 applied to `" with the textual content (given by the `toString` method) of the `Object` received as parameter, in the case of `Lambda1`;
- a `String` that concatenates "`Lambda2 applied to `" with the textual content (given by the `toString` method) of the `Object` received as parameter, in the case of `Lambda2`.

Lambda1:
```
public class Lambda1{
      public Object apply(Object x) {
            return "Lambda1 applied to "+x.toString();
      }
}
```

Lambda2:
```
public class Lambda2{
      public Object apply(Object x) {
            return "Lambda2 applied to "+x.toString();
      }
c)}
```

d) **[0.5 marks]** Consider another interface, called `ILogical`, which represents objects with the ability to apply one of two possible *lambdas*. In OOP the `ILogical` interface defines a method `select` with three parameters: two `ILambda`'s and an `Object`. The `select` method calls either the 1st or 2nd `ILambda apply` method with the given `Object`, and returns the `Object` returned in the call to the `apply` method. Define the `ILogical` interface.

ILogical:
```
public interface ILogical {
      public Object select(ILambda tlambda, ILambda flambda, Object x);
}
```

Name:	Number:

e) **[1.5 marks]** Define two concrete classes, called `True` and `False`, which implements `ILogical` and satisfy the following specification. When

- `b.select(new Lambda1(), new Lambda2(), "hello");`

is executed with a variable `b` of type `ILogical`, the return value will be

- `"Lambda1 applied to hello"`, when `b` is of type `True`, and
- `"Lambda2 applied to hello"`, when `b` is of type `False`.

f) Your code for `True` and `False` may not contain any `String` literals. Exemplify the call of `select` over a `True` object in a `main` method within a `Main` class.

True:

```
public class True implements ILogical {
   public Object select(ILambda tlambda, ILambda flamdba, Object x){
      return tlambda.apply(x);
   }
}
```

False:

```
public class False implements ILogical {
   public Object select(ILambda tlambda, ILambda flamdba, Object x){
      return flambda.apply(x);
   }
}
```

Main:

```
public class Main {
   public static void main(String[] args) {
      ILogical b = new True();
      b.select(new Lambda1(), new Lambda2(), "hello");
   }
}
```

3 – Fill the answers of multiple choice questions in the following table (use only capital letters). If you want to correct your answer scratch out and write the correct answer. Each correct question is marked 0.5 points. A question not answered is marked 0 points, whereas a wrong answer discounts 0.2 points.

Question	a)	b)	c)	d)	e)
Answer	B	D	C	C	C

a) **[0.5 marks]** What is the value of the static attribute `x` and the instance attribute `y` after a second object of type `X` being created?

```
class X {
      static int x=0;
      int y=-1;
      { y=1; }
      public X() { x=x+y; }
}
```

A. `x=1 and y=1`
B. `x=2 and y=1`
C. `x=-1 and y=-1`
D. `x=-2 and y=-1`
E. None of the above

Name:	Number:

b) **[0.5 marks]** Assume that we are importing `java.util.Arrays`. What is it printed to the terminal?

```
int[][] tiVar1 = new int[3][3];
int[][] tiVar2 = new int[3][3];
System.out.print(tiVar1==tiVar2);
System.out.print(tiVar1.equals(tiVar2));
System.out.print(Arrays.equals(tiVar1,tiVar2));
System.out.print(Arrays.deepEquals(tiVar1,tiVar2));
```

A. `true true true true`
B. `false true true true`
C. `false false true true`
D. `false false false true`
E. None of the above

c) **[0.5 marks]** What is it printed to the terminal?

```
class X{
      public int xpto(){return 5;}
      public static int foo() {return 15;}
      void test(){
            System.out.print(foo());
            System.out.print(xpto());
            System.out.print(((Y)this).foo());
            System.out.print(((X)this).xpto());
      }
}
class Y extends X{
      public int xpto(){return 10;}
      public static int foo() {return 20;}
      public static void main(String[] args){
            new Y().test();
      }
}
```

A. `15 5 20 5`
B. `20 10 20 10`
C. `15 10 20 10`
D. `20 10 20 5`
E. None of the above

d) **[0.5 marks]** An `HashSet<?>` denotes:

A. `HashSet<Object>`
B. `HashSet<? super Object>`
C. `HashSet<? extends Object>`
E. None of the above

e) **[0.5 marks]** What kind of exception `NullPointerException` is?
A. `Error`
B. `Checked exception`
C. `Unchecked exception`
D. None of the previous

Name:	Number:

4 – [1.0 marks] Consider the following program, in which the class `IntPair` represents a pair of integers.

```
public class IntPair {
      private final int first;
      private final int second;
      public IntPair(int first, int second) {
            this.first = first;
            this.second = second;
      }
      @Override public boolean equals(Object o) {
            if (!(o instanceof IntPair)) return false;
            IntPair ip = (IntPair) o;
            return ip.first==first;
     }
     @Override public int hashCode() {
            return 31*first;
     }
     public static void main(String[] strings){
            Set<IntPair> s = new HashSet<IntPair>();
            for (int i=0; i<10; i++)
                  for (int j=0;j<10;j++)
                        s.add(new IntPair(i,j));
            System.out.println(s.size());
            System.out.println(s.contains(new IntPair(0,100)));
     }
}
```

We might expect the program to print 100 and false, but after running this program we find out that it prints not 100 and false but 10 and true! Spot the bug and correct it. Justify your answer. Note: Sets cannot contain duplicates.

The problem is that the method equals only relies on the first integer to compare objects of type IntPair. To correct the bug we need to make the equals method also rely on the second integer of the pair of IntPairs. Moreover, since the hashCode method should be defined consistently with equals, we also need to change the hashCode method.

```
@Override public boolean equals(Object o) {
      if (!(o instanceof IntPair)) return false;
            IntPair ip = (IntPair) o;
            return ip.first==first && ip.second==second;
}

@Override public int hasCode() {
      return 31*first+second;
}
```

Name:	Number:

Parte III -- XML (1 mark)

5 – [1.0 marks] Consider the problem presented in Part I and, particularly, problem 1b), that is, consider n Home Control System with four devices associated with one dinner room. Informally, present a XML document, with XML elements and attributes, to store all information at hand. Be especially careful with the representation of the one-to-many relationship between house rooms and home devices.

```
<HomeControlSystem>
      <Devices>
            <TV ID ="1" roomREF="100"/>
            <HouseLights>
                  <EmergencyLight ID="2" roomREF="100"/>
                  <RoomLight ID="3" roomREF="100"/>
                  <RoomLight ID="4" roomREF="100"/>
            </HouseLights>
      </Devices>
      <HouseRooms>
            <DinnerRoom ID="100" name="DN1">
                  <HomeDeviceRefs>
                        <HomeDeviceRef deviceREF="1"/>
                        <HomeDeviceRef deviceREF="2"/>
                        <HomeDeviceRef deviceREF="3"/>
                        <HomeDeviceRef deviceREF="4"/>
                  </HomeDeviceRefs>
            </DinnerRoom>
      </HouseRooms>
</HomeControlSystem>
```

Parte IV -- C++ (1 mark)

6 – [1.0 marks] Implement in C++ a `swap` method, with parameters passed by reference, that swaps the value of the two parameters of type `int` (parameters should be references to `int`). Exemplify its usage by swapping the values of two `int` variables.

```
void swap(int& i, int& j) {
      int tmp = i;
      i = j;
      j = tmp;
}
```

```
int k = 2, r = 3;
swap(k,r);
```