Name: Number:

Object Oriented Programming 2008/09

Final Exam
July 24th, 2009

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, plus 30 minutes of tolerance.
- 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	6	2.5
	4	Java miscellaneous	7	1.0
III	5	XML	8	1.0
IV	6	C++	8	1.0
Total				**10.0**

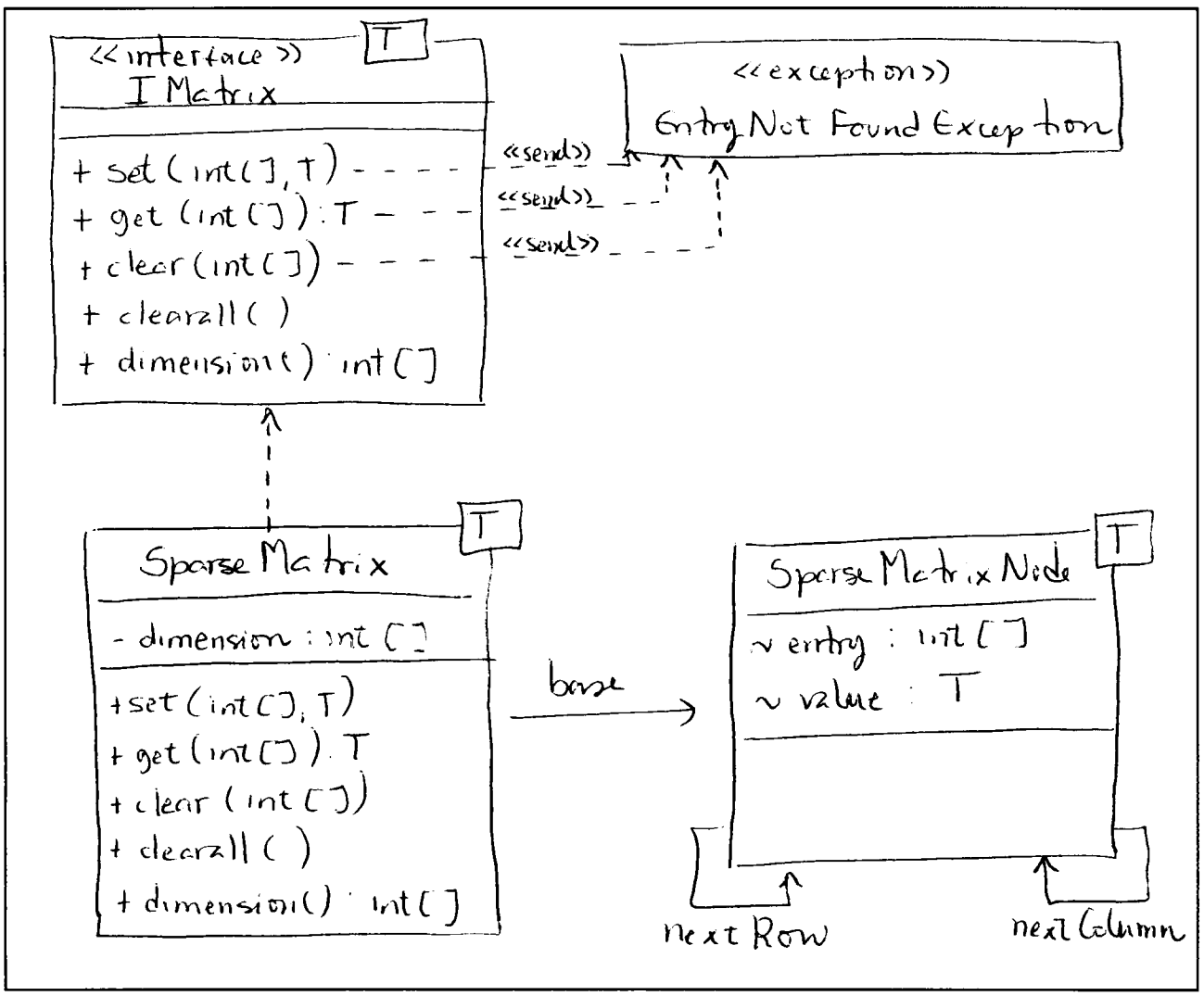
Name: Number:

Part I -- UML (1.5 marks)

1 – We intend to develop a package to make available usual operations over generic matrixes. In this context any matrix implementation should always offer the following methods: (i) the set method, that sets an entry <i,j> with a value, both received by parameter; (ii) the get method, that returns the value stored in entry <i,j> received by parameter; (iii) the clear method, that sets to zero an entry <i,j> received by parameter; (iv) the clearall method, that clear all entries; and (v) the dimension method, that returns the dimension of the matrix. Entries needed by methods set, get and clear, and the return type of the dimension method, should be received in an array form. The methods set, get and clear should throw `EntryNotFoundException` whenever one tries to operate over an entry that does not exist.

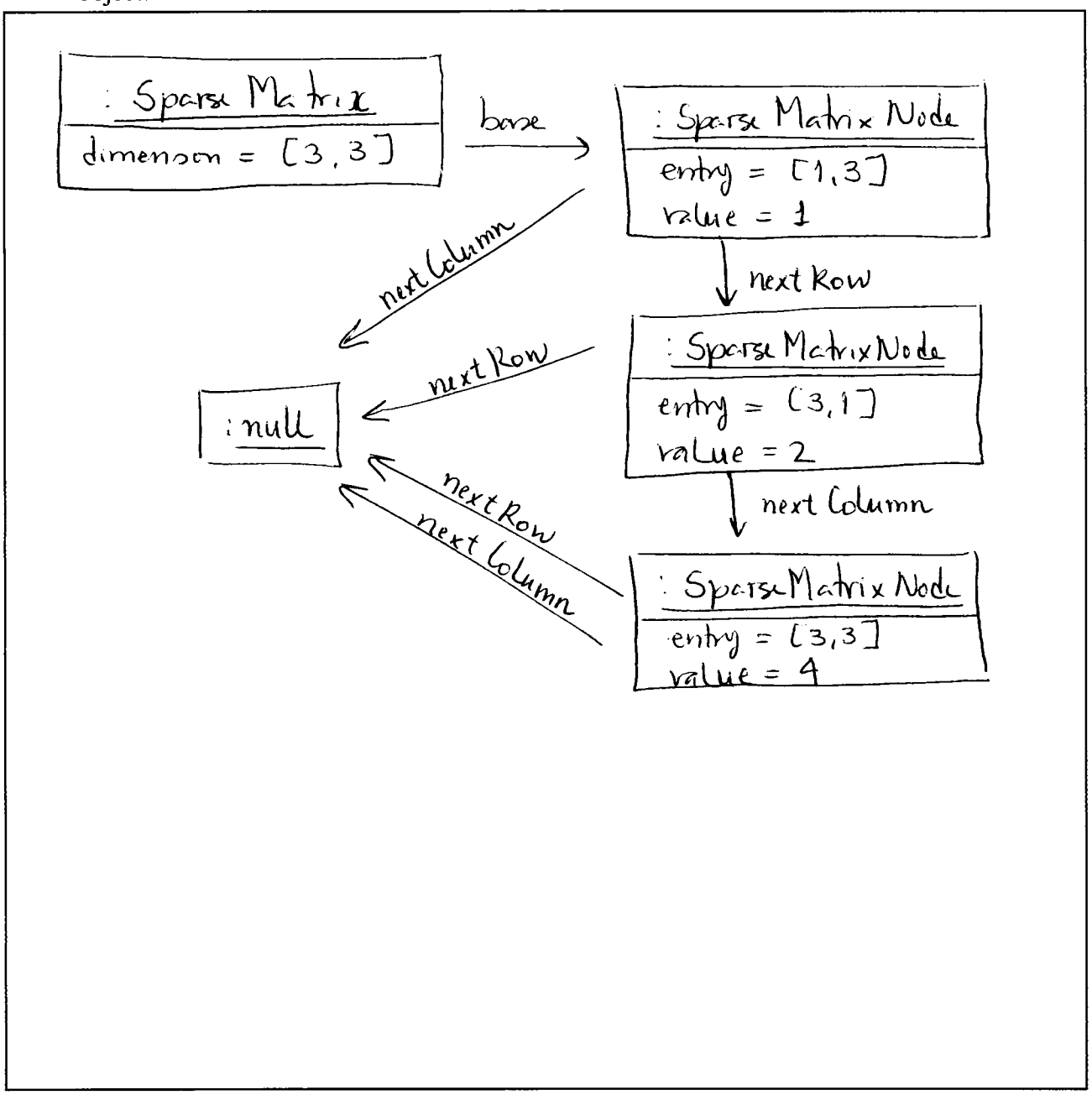
Moreover, we intend to offer a particular implementation of a generic matrix: a sparse generic matrix. A sparse matrix is one for which almost all entries are zeros. Taking this into account we intend to develop a particular implementation where only the non-zero values are stored in memory. For that purpose, consider a doubly linked list, where one link represents the next column and the other link represents the next row. Note that in the same row, the next column points to the next filled entry in the same row. If there is no such column then next column should points to null. The next row works differently. It always points to the first filled entry in the next non-empty row. If there is no such row then next row should point to null.

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 we have in memory a sparse matrix of integers with three elements: the value 1 at entry <1,3>, the value 2 at entry <3,1>, and the value 4 at entry <3,3>. This matrix is of dimension [3,3]. Hint: null is a special object.



Name: Number:

Parte II -- Java (6.5 marks)

2 – One of the problems related to the implementation of linked lists is to deal with the special case of an empty list. For instance, when adding a node to an empty list we need to update the reference to the base of the list. The same happens when removing a node from a singleton list. To overcome this problem it is usual to consider a spurious base, usually of the same type of list nodes, which is created when the list is created and it is never deleted unless the list is deleted. This spurious base points to the first node of the list (and stores a null value).

a) **[1.0 marks]** Define a generic spurious base linked list. Call this list `SBLinkedList` and its nodes `SBLinkedListNode`. For both classes, consider only attributes and constructors needed to construct an empty list and to solve questions 2b) and 2c).

`SBLinkedList`:

```
public class SBLinkedList<T> {

      private int length;
      private NodeSBLinkedList<T> sbase;

      public SBLinkedList() {
            length = 0;
            sbase = new NodeSBLinkedList<T>();
      }
}
```

`SBLinkedListNode`:

```
public class SBLinkedListNode<T> {

      T element;
      SBLinkedListNode<T> next;

      public SBLinkedListNode() {}
      public SBLinkedListNode(T element, SBLinkedListNode<T> next) {
            this.element = element;
            this.next = next;
      }
}
```

b) **[0.5 marks]** Implement the `add` method in `SBLinkedList` class. This method should receive the element do add to the list and returns nothing.

```
public void add(T elem){

      sbase.next = new SBLinkedListNode<T>(elem,sbase.next);
      length++;
}
```

c) **[0.7 marks]** Implement the `remove` method in `SBLinkedList` class. This method should receive the element to remove from the list and returns true if such element was actually found and removed, and false otherwise. Consider removing only one element.

```
public boolean remove(T elem) {

      if (length==0) return false;

      SBLinkedListNode<T> previous = sbase;
      int i = 0;
      while (i<length) {
            if (previous.next.element.equals(elem)) {
                  previous.next = previous.next.next;
                  length--;
                  return true;
            }
            previous = previous.next;
            i++;
      }
      return false;
}
```

d) **[0.8 marks]** Implement the `equals` method to return true if two lists are deeply the same (the same elements, node by node).

```
@Override
public boolean equals(Object obj) {

      // Standard equals() tests...
      if (this == obj) return true;
      if (!(obj instanceof SBLinkedList<?>)) return false;

      // Now do deep compare
      @SuppressWarnings("unchecked")
      final SBLinkedList<T> other = (SBLinkedList<T>)obj;
      if (this.length != other.length) return false;
      if (this.length == 0) return true;
      SBLinkedListNode<T> thisnode = this.sbase.next,
                          othernode = other.sbase.next;
      int i=0;
      while (i<this.length) {
            if (!thisnode.element.equals(othernode.element))
                  return false;
            thisnode = thisnode.next;
            othernode = othernode.next;
            i++;
      }
      return true;

}
```

3 – Fill the answers of multiple choice questions in the table (use only capital letters). If you want to correct your answer scratch out and write the correct answer. Each question of multiple choice 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	C	C	C	C	B

a) **[0.5 marks]** Assume classes `A` and `B` are defined at the same package. What is the value of the attributes of `B` after a `new B(10)`?

```
public class A {
      String str;
      public A() {str="Xpto";}
      public A(String str) {this.str=str;}

}
public class B extends A {
      int num;
      public B() {super("Hello");}
      public B(int num) {this();num++;}
}
```

A. `str="Xpto" num=0`
B. `str="Xpto" num=11`
C. `str="Hello" num=0`
D. `str="Hello" num=1`
E. None of the above

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

```
Integer i = new Integer(2);
Integer j = new Integer(2);
System.out.print(i==j);
System.out.print(i.equals(j));
```

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

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

```
class X{
      string str = "xpto";
      int xpto(){return 5;}
}
class Y extends X{
      string str = "ypto";
      int xpto(){return 10;}
      void test(){
            X x = (X) this;
            System.out.print(((X)this).xpto());
            System.out.print(((X)this).str);
            System.out.print(super.xpto());
            System.out.print(super.str);
      }
      public static void main(String[] args){
            new Y().test();
      }
}
```

A. `10 ypto 5 ypto`
B. `5 xpto 5 xpto`
C. `10 xpto 5 xpto`
D. `10 ypto 5 xpto`
E. None of the above

d) **[0.5 marks]** Assume assertions are on. What an assertion throws when it fails?

A. `Exception` B. `RuntimeException` C. `Error` D. `IOException` E. None of the previous

e) **[0.5 marks]** Which interface one should implement to define a natural order?

A. `Comparator` B. `Comparable` C. `Iterator` D. `Iterable` E. None of the previous

4 – [1.0 marks] Sometimes it is useful to require assertions to be always enabled for a particular class at runtime. Implement a class called Universe that throws an exception upon class loading when assertions are disabled.

```
public class Universe {

    static {
        boolean hasAssertEnabled = false;
        assert hasAssertEnabled = true; // rare - an intentional side effect!
        if (!hasAssertEnabled)
            throw new RuntimeException("Asserts must be enabled!");
    }

    //Other class definition goes here.
}
```

Name: Number:

Parte III -- XML (1 mark)

5 – [1.0 marks] Consider the following XML document where CONTACTREF is a reference to another CONTACT in CONTACTS. Present a DTD to validate the given XML document.

```
<CONTACTS>
    <CONTACT CONTACTNUM = "2">
        <NAME>John Smith</NAME>
        <EMAIL>jsmith@xpto.com</EMAIL>
    </CONTACT>
    <CONTACT CONTACTNUM = "1" CONTACTREF = "2">
        <NAME>Luke Smith</NAME>
        <EMAIL>lsmith@ypto.com</EMAIL>
    </CONTACT>
</CONTACTS>
```

```
<?xml version = "1.0" encoding="UTF-8" standalone = "yes"?>
<!DOCTYPE CONTACTS [
        <!ELEMENT CONTACTS (CONTACT*)>
        <!ELEMENT CONTACT (NAME, EMAIL)>
        <!ELEMENT NAME (#PCDATA)>
        <!ELEMENT EMAIL (#PCDATA)>
        <!ATTLIST CONTACT CONTACTNUM ID #REQUIRED>
        <!ATTLIST CONTACT CONTACTREF IDREF #IMPLIED>
]>
```

Parte IV -- C++ (1 mark)

6 – [1.0 marks] There is a problem in the following block of code. Give two alternative solutions (preferably C++ code, but C++ pseudo-code, English or Portuguese written is also accepted).

```
void f (int *p) {...}

int main(void){
      const int a=10;
      const int *b=&a;
      f(b);
}
```

```
ALTERNATIVE 1:

void f (const int *p){...}

int main(void){
      const int a=10;
      const int *b=&a;
      f(b);
}
```

```
ALTERNATIVE 2:

void f (int *p) {...}

int main(void){
      const int a=10;
      const int *b=&a;
      int *c = const_cast<int*>(b);
      f(c);
}
```