CSC207H1 S 2015 Midterm Test
Duration — 50 minutes
Aids allowed: none

Student Number: |_|_|_|_|_|_|_|_|_|_|_|

Last Name: _____ **First Name:** _____

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above, **write your name on the back of the test**, and read the instructions below.)
Good Luck!

This midterm consists of 4 questions on 8 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*

- Legibly write your name and student number on this page. Legibly write your name on the back page of this exam.
- If you use any space for rough work, indicate clearly what you want marked.
- In all programming questions you may assume all input is valid.
- You do not need to write Javadoc.
- You may use a pencil; however, work written in pencil will not be considered for remarking.

1: _____/ 8

2: _____/ 8

3: _____/13

4: _____/ 5

TOTAL: _____/34

Question 1. [8 MARKS]

Anish and Zifei are working together on Phase I of their CSC207 project. They share a Subversion repository located at `http://markus.cdf.toronto.edu/csc207-2015-01/superteam`. The current version in the repository is **revision number 5**. The repository contains two files, `todo.txt` and `data.txt`, with the following contents:

`todo.txt`

```
CRC
demo
submit
```

`data.txt`

```
List
Map
```

Q: Zifei just bought a new laptop and wants to get a local copy of the repository on it. What command does she need to execute?

```
svn checkout http://markus.cdf.toronto.edu/csc207-2015-01/superteam
```

Suppose now that Anish does some work in his local copy. He then runs `svn status`:

```
anish$ svn status
D       todo.txt
?       walkthrough.txt
M       data.txt
```

Q: What sequence of commands does Anish need to issue so that the changes made locally (including any newly created files) are part of the repository?

```
svn add walkthrough.txt
svn commit -m "Removing todo list; adding walkthrough; updating data"
```

Q: After the commands above are issued, what is the revision number in:

- Anish's copy? `6`
- Zifei's copy? `5`
- the repository? `6`

Q: What sequence of command(s) does Zifei need to issue so that her local copy is at the latest revision?

`svn update`

Anish and Zifei both have the latest revision in their local copies and there are no local changes. Now, Anish and Zifei both edit their local copies of `data.txt`:

`Anish: data.txt`

Set
List
Map

`Zifei: data.txt`

List
Map
Queue

Q: Assuming the changes they made do not conflict, what sequence of commands do Anish and Zifei need to issue so that their changes become part of the respository and so that they both have the most recent version of `data.txt`? For each command, state who needs to execute it, Anish or Zifei.

Anish: svn commit -m 'added a line'
Zifei: svn update
Zifei: svn commit -m 'added a line'
Anish: svn update

The file `data.txt` should now have these contents:

`data.txt`

Set
List
Map
Queue

Question 2. [8 MARKS]

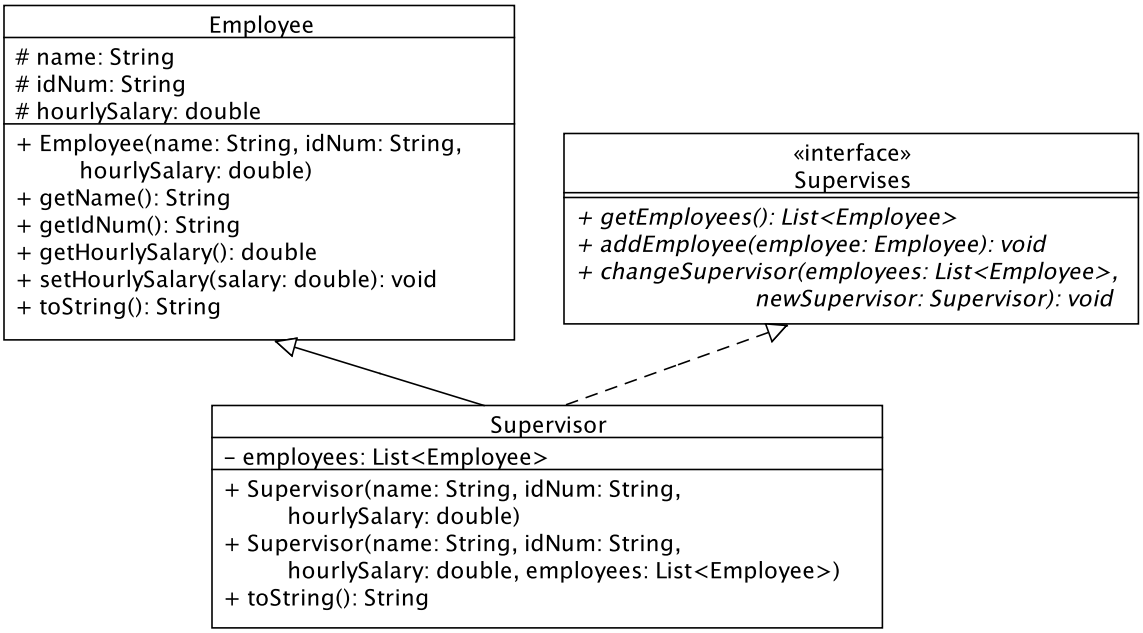
Consider this `main` method:

```
public static void main(String[] args) {

    System.out.println(Pair.getNumberOfPairs()); // prints: 0

    Pair<Integer> pairOne = new Pair<>(new Integer(3), new Integer(4));
    System.out.println(Pair.getNumberOfPairs()); // prints: 1

    Pair<String> pairTwo = new Pair<>("hi", "bye");
    System.out.println(Pair.getNumberOfPairs()); // prints: 2

    System.out.println(pairOne.toString()); // prints: (3, 4)
    System.out.println(pairTwo.toString()); // prints: (hi, bye)

    pairOne.setX(new Integer(8));
    System.out.println(pairOne.getX()); // prints: 8
}
```

Implement class `Pair` so that the `main` method above compiles, runs, and produces the output shown. You should assume the `main` method is in class `Pair`.

```
public class Pair<T> {

    private T x;
    private T y;
    private static int numberOfPairs = 0;

    public Pair(T x, T y) {
        this.x = x;
        this.y = y;
        numberOfPairs++;
    }

    public static int getNumberOfPairs() {
        return numberOfPairs;
    }

    public void setX(T x) {
        this.x = x;
    }
    public T getX() {
        return x;
    }

    public String toString() {
        return "(" + x + ", " + y + ")";
    }
}
```

Question 3. [13 MARKS]



Assume that interface `Supervises` and class `Employee` have been implemented. Implement class `Supervisor` according to the UML diagram. Here is information about `Supervisor`'s methods:

- Method `toString` returns a string that includes the string produced by `Employee`'s `toString` and a string representation of this `Supervisor`'s list of employees.
- Method `changeSupervisor` checks whether each `Employee` in the given list of employees is supervised by this `Supervisor` and, if so, removes the `Employee` from this `Supervisor`'s list of employees and adds the `Employee` to the new `Supervisor`'s list of employees.

```
public class Supervisor extends Employee implements Supervises {

    private List<Employee> employees;

    public Supervisor(String name, String idNum, double dailySalary) {
        super(name, idNum, dailySalary);
        employees = new ArrayList<>();
    }
    public Supervisor(String name, String idNum, double dailySalary, List<Employee> employees) {
        super(name, idNum, dailySalary);
        this.employees = employees;
    }

    public void addEmployee(Employee employee) {
        employees.add(employee);
    }

    @Override
    public List<Employee> getEmployees() {
        return employees;
    }

    @Override
    public void changeSupervisor(List<Employee> employees, Supervisor newSuper) {

        for (int i = 0; i < employees.size(); i++) {
            Employee e = employees.get(i);
            if (this.employees.contains(e)) {
                this.employees.remove(this.employees.indexOf(e));
                newSuper.employees.add(e);
            }
        }

        // Another sample solution:
//      for (Employee e: employees) {
//          if (this.employees.contains(e)) {  // "this." is required here
//              this.employees.remove(e);
//              newSuper.employees.add(e);
//          }
//      }
    }

    public String toString() {
        return super.toString() + employees;
    }
}
```

Question 4. [5 MARKS]

The following code compiles without errors:

```
public class Top {
  public int value = 1;

  public void printValue() {
    System.out.println(value);
  }
}
```

```
public class Mid extends Top {
  public void printValue() {
    System.out.println(value);
  }
}
```

```
public class Bottom
  extends Mid {
    public int value = 3;
}
```

Suppose we have a `main` method that contains this code segment:

```
Top myVar = new Bottom();
```

(a) [1 MARK] Which ***variable*** is accessed in the expression `myVar.value`?

- [√] The instance variable `value` in class `Top`.
- [ ] The instance variable `value` in class `Bottom`.
- [ ] None of the above; this expression is illegal.

(b) [1 MARK] Which ***variable*** is accessed in the expression `((Mid) myVar).value`?

- [√] The instance variable `value` in class `Top`.
- [ ] The instance variable `value` in class `Bottom`.
- [ ] None of the above; this expression is illegal.

(c) [1 MARK] Which ***variable*** is accessed in the expression `((Bottom) myVar).value`?

- [ ] The instance variable `value` in class `Top`.
- [√] The instance variable `value` in class `Bottom`.
- [ ] None of the above; this expression is illegal.

(d) [1 MARK] If the method `myVar.printValue()` is called, which ***method*** is executed?

- [ ] The method `printValue` in class `Top`.
- [√] The method `printValue` in class `Middle`.
- [ ] None of the above; this method call is illegal.

(e) [1 MARK] If the method `((Bottom) myVar).printValue()` is called, which ***method*** is executed?

- [ ] The method `printValue` in class `Top`.
- [√] The method `printValue` in class `Middle`.
- [ ] None of the above; this method call is illegal.

Last Name: _____ **First Name:** _____

```
interface List<E> extends Collection<E>, Iteratable<E>:
    // An ordered Collection. Allows duplicate items.

    boolean add(E elem)
      Appends elem to the end of this List.
    void add(int i, E elem)
      Inserts elem at index i if this List.
    boolean contains(Object o)
      Returns true iff this List contains o.
    E get(int i)
      Returns the item at index i of this List.
    int indexOf(Object o)
      Returns the index of the first occurrence of o, or -1 if not in this List.
    boolean isEmpty()
      Returns true iff this List contains no elements.
    E remove(int i)
      Removes the item at index i of this List and returns the item.
    int size()
      Returns the number of elements in this List.

class ArrayList<E> implements List<E>
```

SVN commands

```
svn add [PATH]
svn checkout URL
svn commit [PATH] -m MESSAGE
svn delete [PATH]
svn list [PATH]
svn status [PATH]
  Meaning of output:
    A: item is scheduled for addition.
    D: item is scheduled for deletion.
    M: item has been modified.
    ?: item is not under version control.
svn update [PATH]
```

Total Marks = 34

End of Solutions