Université d'Ottawa
Faculté de génie

École d'ingénierie et de
technologie de l'information

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information
Technology and Engineering

Introduction to Computing II (ITI 1121)
Midterm Examination

Instructor: Marcel Turcotte

March 2014, duration: 2 hours

Identification

Surname: _____ Given name: _____

Student number: _____ Section (A or B): _____

Instructions

1. This is a closed book examination.
2. No calculators, electronic devices or other aids are permitted.
   (a) Any electronic device or tool must be shut off, stored and out of reach.
   (b) Anyone who fails to comply with these regulations may be charged with academic fraud.
3. Write your answers in the space provided.
   (a) Use the back of pages if necessary.
   (b) You may not hand in additional pages.
4. Write comments and assumptions to get partial marks.
5. Do not remove the staple holding the examination pages together.
6. Beware, poor hand writing can affect grades.
7. Wait for the start of the examination.

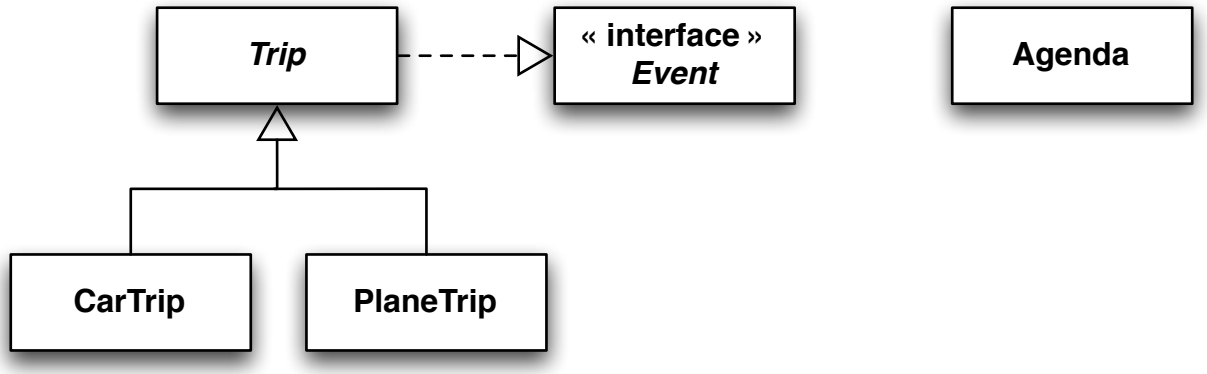
Marking scheme

Question	Maximum	Result
1	35	
2	15	
3	15	
Total	**65**	

All rights reserved. No part of this document may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without prior written permission from the instructor.

Question 1 (35 marks)

A recent study by the Government of Akalee showed that 42% of the country's civil servants had missed at least three business trips they were supposed to take over a period of one year. As a means of dealing with this high level of disorganisation amongst its staff, your company was commissioned with the task of implementing a business trip agenda for each and every state employee. The UML diagram below gives you an overview of the application.



Follow all the instructions below, and make sure to include constructors, as well as getters and setters for all the attributes.

- Implement the interface **Event**. It declares two methods, **getStart()** and **getEnd()**, both returning a value of type **Date**. (5 marks)
- Write the implementation of the abstract class **Trip**. It implements the characteristics that are common to its sub-classes, here **CarTrip** and **PlaneTrip**. (8 marks)
  - **Trip** implements the interface **Event**.
  - All the trips have a start and end date, as well as a departure location and a destination. Their initial values are passed to the constructor.
  - **Trip** declares an abstract method called **checklist**, which checks whether all the conditions for a trip are satisfied. If they are, it returns **true**. Otherwise, it returns **false**.
- Implement the class **CarTrip** (6 marks)
  - **CarTrip** has a constructor which, in addition to the default parameters of **Trip**, also assigns values to **isFull** (indicates whether or not the gas tank is full), **breaksChecked** (**true** or **false**), and the number of passengers (an integer).
  - **CarTrip** implements the method **checklist**. It returns **true** if the gas tank is full, the breaks were checked, and the number of people traveling is 4 or less, which is the capacity of the car.
- Implement the class **PlaneTrip** (6 marks)
  - **PlaneTrip** has a constructor which, in addition to the default parameters of **Trip**, also assigns values to **ticketIssued** (**true** or **false**), **passportValid** (**true** or **false**), **numberCheckedLuggage** (an integer), and **numberCarryOn** (an integer).

  - **PlaneTrip** implements the method **checklist**. It returns **true** if the ticket was issued, the passport is valid, the number of checked items is two or less, and the number of carry-on items is one or less.

- Implement the class **Agenda**. Similarly to Assignment #1 Question #2, **Agenda** uses a fixed-size array to store elements. Give all the necessary class and instance variables, as well as a constructor. For simplicity, herein, you will only implement one method, **add**. (10 marks)

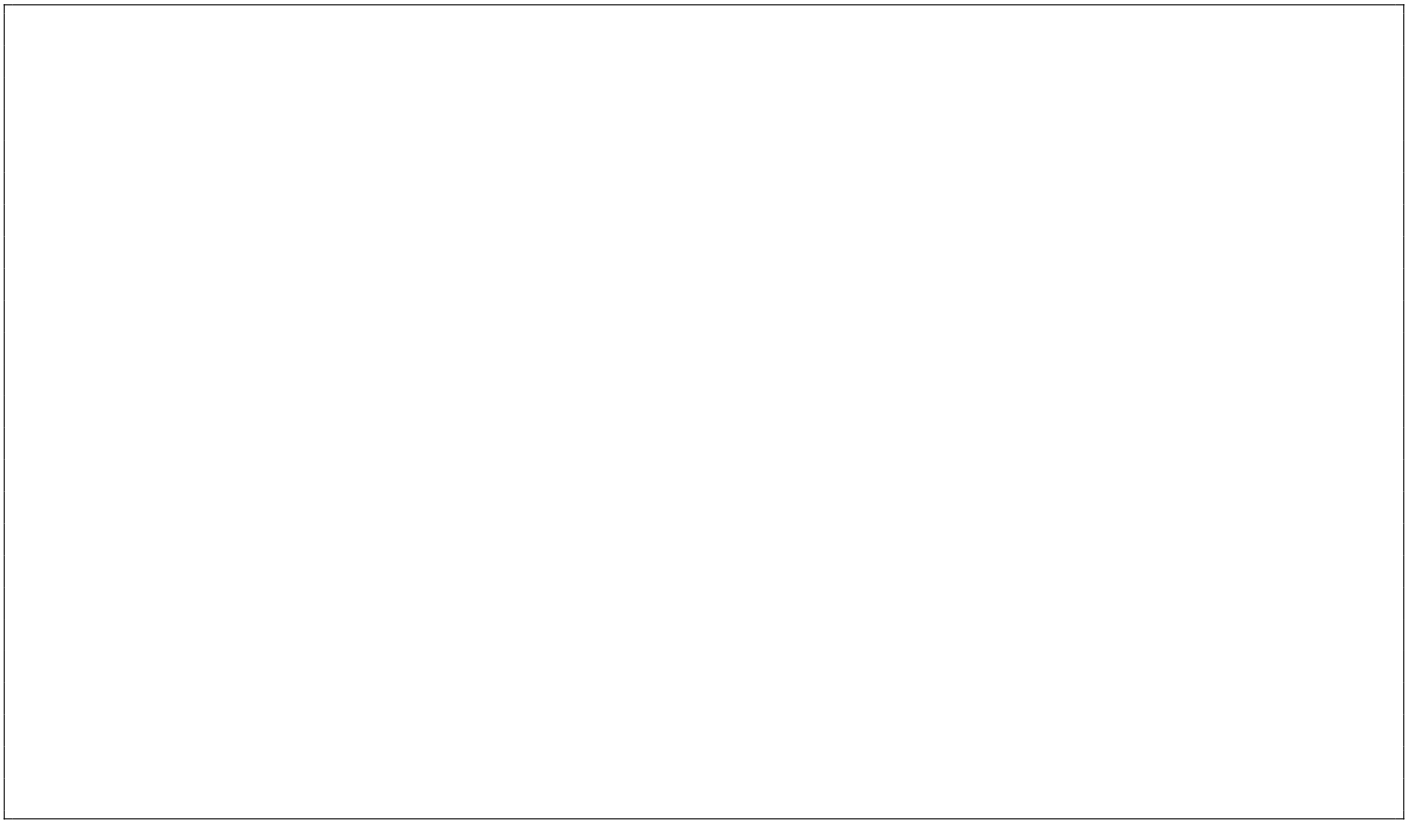
  - **boolean add(Event e):** adds an event to the agenda. Events are kept in increasing order of start time. The method returns **true** if the event was added to the collection, and **false** otherwise.

In your implementation, use the class **java.util.Date** to represent the start and end time of events. This class implements the interface **Comparable**, which declares a method **int compareTo(Date other)**. In the example below, an object of the class **DateFormat** is used to create an object of the class **Date** from a **String** representation.

```
Agenda agenda;
Trip[] trips;
DateFormat df;
Date start, end;

agenda = new Agenda(10);

trips = new Trip[4];

df = DateFormat.getDateInstance(DateFormat.SHORT, Locale.CANADA);

start = df.parse("09/03/2014"); end = df.parse("14/03/2014");
trips[0] = new CarTrip(start, end, "Okola", "Bulni", true, true, 1);

start = df.parse("22/11/2014"); end = df.parse("04/12/2014");
trips[1] = new PlaneTrip(start, end, "Okola", "Klappo", true, false, 3, 1);

start = df.parse("15/06/2014"); end = df.parse("17/06/2014")
trips[2] = new PlaneTrip(start, end, "Okola", "Pilne", true, true, 2, 1);

start = df.parse("05/02/2014"); end = df.parse("13/02/2014");
trips[3] = new PlaneTrip(start, end, "Okola", "Talba", true, true, 1, 1);

for (int i=0; i<trips.length; i++) {
  if (trips[i].checkList()) {
    agenda.add(trips[i]);
  }
}
```

Following the execution of the above statements, the agenda contains three events, in the following order, the PlaneTrip starting on 05/02/2014, followed by the CarTrip starting on 09/03/2014, followed by the PlaneTrip starting on 15/06/2014.

Event



Trip

CarTrip

PlaneTrip

Agenda

Question 2 (15 marks)

For this question, you must implement the class (static) method **boolean isSkipped(Stack s1, Stack s2)**.

- The method **isSkipped** returns **true** if and only if the stacks designated by **s1** and **s2** contain the same elements, in the same order, but with elements 2,4,6,... missing in **s2**. Given **s1** designating a stack containing the elements 1,2,3,4,5,6,7, where 7 is the top element, and **s2** designating a stack containing the elements 1,3,5,7, where 7 is the top element, **isSkipped(s1, s2)** returns **true**.
- Both stacks, designated by **s1** and **s2**, must remain unchanged following a call to the method **isSkipped**. Specifically, given **s1** and **s2**, two reference variables designating stacks, following the call **isSkipped(s1, s2)**, **s1** contains the same elements, in the same order, as it did before the call to **isSkipped**. Similarly, **s2** contains the same elements, in the same order, as it did before the call to **isSkipped**.
- You can assume that both, **s1** and **s2**, will not be **null**.
- An empty stack is considered the skipped version of another empty stack.
- The parameters of the method **isSkipped** are of type **Stack**, which is an interface.

For this question, there is an interface named **Stack**:

```
public interface Stack {
  public abstract void push(int item);
  public abstract int pop();
  public abstract boolean isEmpty();
}
```

- Notice that the parameter of the method **push** and the return value of the method **pop** are of type **int**.
- Assume the existence of **DynamicStack**, which implements the interface **Stack**. It has one constructor and its signature is **DynamicStack()**.
- You cannot use arrays to store temporary data.
- You must use objects of the class **DynamicStack()** to store temporary data.
- You do not know anything about the implementation of **DynamicStack**. In particular, you do not know if it uses an array or not.
- You can assume that **DynamicStack** can store an arbitrarily large number of elements.

Write your answer on the next page.

```
public class Q2 {
  public static boolean isSkipped(Stack s1, Stack s2) {









  } // End of isSkipped
} // End of Q2
```

Question 3 (15 marks)

A. True or False Questions (5 marks)

(a) The program below will print 19.
True or **False**.

```
public class Amount {
  private int value = 0;

  public int getValue() {
    return value;
  }

  public void setValue(int base) {
    value = base;
  }

  public void incr(int increment) {
    value = value + increment;
  }
}
public class Test {
  public static void main(String[] args) {
    Amount pocketMoney = new Amount();
    pocketMoney.setValue(17);
    pocketMoney.incr(5);
    pocketMoney.value = pocketMoney.value - 3;
    System.out.println(pocketMoney.getValue());
  }
}
```

(b) The program below will print **true**.
True or **False**.

```
public class Test {
  public static void main(String [ ] args) {
    Integer a , b ;
    a = new Integer(35);
    a++;
    b = new Integer(36);
    System.out.println( a != b );
  }
}
```

(c) An instance variable cannot be used (referenced) in a class (static) method.
True or **False**.

(d) In the class **OlympicTime** below, the statement **super(h,m)** could be omitted since Java automatically adds that call in the constructor of a sub-class.
True or **False**.

(e) In the class **OlympicTime** below, "**this**" can be omitted in the two statements in which it occurs.
True or **False**.

```
public class Time extends Object {
    protected int hours;
    protected int minutes;
    public Time(int h, int m) {
        hours = h;
        minutes = m;
    }
}

public class OlympicTime extends Time {
    private int seconds;
    private int milliseconds;
    public OlympicTime(int h, int m, int v1, int v2) {
        super(h, m);
        this.seconds = v1;
        this.milliseconds = v2;
    }
}
```

B. Multiple Choice Questions (6 marks)

(a) Which of the following statements is or are correct?

i. An interface can contain concrete methods, as long as it contains at least one abstract method.

ii. An abstract class can be extended by several sub-classes.

iii. A class can implement several interfaces.

A. i

B. i and ii

C. i and iii

D. only iii

E. ii and iii

(b) What is the value of the following postfix expression? 5 2 − 6 × 3 1 − /

i. 0

ii. 9

iii. 2

iv. 11

v. not a well formed postfix expression

(c) Given a method having a formal parameter of type **A**, _____ allows for the type of the actual parameter for the call to be **B**, where **B** is a subclass of **A**.

- polymorphism
- inheritance
- method overloading
- generic types
- none of the above

C. Following the guidelines presented in class, as well as the lecture notes, draw the memory diagrams for all the objects, the local variables, and parameter of the method **Basket.main** following the execution of the statement "**bskt2.items[1].setPrice(8);**". (4 marks).

```
public class Food {
    private String item = null;
    private double price = 0.0;

    public Food(String item, double price) {
        this.item = item;
        this.price = price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
}

public class Basket {
    private Food[] items;

    public Basket() {
        items = new Food[2];
        items[0] = new Food("Bread", 4.0);
        items[1] = new Food("Chocolate", 6.0);
    }

    public static void main(String[] args) {
        Basket bskt1, bskt2;
        bskt1 = new Basket();
        bskt2 = bskt1;
        bskt2.items[1].setPrice(8.0);
        // here
    }
}
```

(blank space)