School of Computer Science - Faculty of Science - McGill University

COMP 202 - Introduction to Computing 1
Final exam

April 29th, 2003, 9:00am - 12:00

Examiners: Ernesto Posse, Yannick Daoudi, Klaus Reinhardt. Associate examiner: Joseph Vybihal

Last name _____

First name _____

Id number _____

Section ☐ 1 (Ernesto Posse) ☐ 2 (Yannick Daoudi) ☐ 3 (Klaus Reinhardt)

Instructions

- No notebooks, textbooks or calculators are allowed in this exam.
- All answers should be written on the exam sheets. Indicate clearly where are the solutions for each question in sections 2 and 3. For section 1 (multiple choice) mark your answer in this front page.
- Read carefully the questions.
- Language translation dictionaries are permitted.
- Answers can be given in either English or French.

Grading

Section 1: Multiple choice — Section 1 mark: / 24

Question	Q1	Q2	Q3	Q4	Q5	Q6
Your answer						
Mark	/ 2	/ 2	/ 2	/ 2	/ 2	/ 2
Question	Q7	Q8	Q9	Q10	Q11	Q12
Your answer						
Mark	/ 2	/ 2	/ 2	/ 2	/ 2	/ 2

Section 2: Problems — Section 2 mark: / 36

Question	Q1	Q2	Q3	Q4	Q5
Mark	/ 6	/ 8	/ 8	/ 6	/ 8

Section 3: Programming — Section 3 mark: / 40

Question	Q1	Q2	Q3	Q4
Mark	/ 10	/ 10	/ 10	/ 10

Total: / 100

Some useful methods that you may or may not need:

Class	Method	Arguments	Return type	Description
String	charAt	int i	char	Returns the i-th char of the given string, starting from 0.
String	length	void	int	Returns the length of the given string.
String	equals	String s	boolean	Returns true if the given string has exactly the same characters as s.
Integer	parseInt	String s	int	Returns the integer corresponding to s if s has only digits as characters or -.
Float	parseFloat	String s	float	Returns the float corresponding to s if s has only digits as characters, a decimal point, or -.

Operator precedence (from higher to lower):

Precedence level	Operator	Operation	Associativity
1	*array*`[`*index*`]`	Array indexing	Left to Right
	object`.`*name*	Attribute and method reference	
	method`(`*parameters*`)`	Parameter evaluation and method invocation	
	`++`	Postfix increment	
	`--`	Postfix decrement	
2	`++`	Prefix increment	Right to Left
	`--`	Prefix decrement	
	`+`	Unary plus	
	`-`	Unary minus	
	`!`	Logical negation (NOT)	
3	`new` *class-name*	Object instantiation	Right to Left
	`(`*type*`)`	Cast (type conversion)	
4	`*`	Multiplication	Left to Right
	`/`	Division	
	`%`	Modulo (remainder)	
5	`+`	Addition	Left to Right
	`+`	String concatenation	
	`-`	Substraction	
6	`<`	Less than	Left to Right
	`<=`	Less than or equal to	
	`>`	Greater than	
	`>=`	Greater than or equal to	
	object `instanceof` *class*	Type membership	
7	`==`	Equals (physical i.e. pointer equality)	Left to Right
	`!=`	Not equal	
8	`&&`	Logical conjunction (AND)	Left to Right
9	`||`	Logical disjunction (OR)	Left to Right
10	`=`	Assignment	Right to Left

1 Multiple choice

Question 1 Which of the following is not a Java reserved word (keyword)?

(A) `this`

(B) `implements`

(C) `word`

(D) `byte`

(E) `protected`

Question 2 What does the keyword "private" mean when applied to a method declaration?

(A) The method cannot be overridden by any derived class.

(B) The method cannot be called by any method in any derived class.

(C) The method cannot be overloaded.

(D) The method cannot be called by any method of a different class.

(E) None of the above

Question 3 Which of the following statements is false:

(A) All objects in a class have the same attributes (data fields, or variables) but the values of those attributes are individual to each object, except for static variables, whose values are shared between all objects in the class.

(B) When a variable is declared, an object is created.

(C) When a (non-static) method is invoked, the method executed depends on the type of the object.

(D) There can be different methods with the same name.

(E) If a method is called for a given object and the object's class does not define that method, the Java Virtual Machine looks for the method in the parent class.

Question 4 Which of the following is true:

(A) Static variables are specific to each object.

(B) A static method can call a non-static method, provided that it explicitly creates an object of the class containing the non-static method.

(C) A static method can use the special reference `this`.

(D) A static method can access non-static variables of its own class.

(E) A non-static method cannot access neither static methods nor static variables.

Question 5 Assume that `BankAccount` is a predefined class and that the declaration `BankAccount[] firstEmpireBank;` has been made. Then the statement `firstEmpireBank = new BankAccount[1000];` reserves memory space for:

(A) A reference variable to the memory that stores all 1000 BankAccount objects.

(B) 1000 reference variables, each of which can point to a single BankAccount object.

(C) 1000 BankAccount objects.

(D) 1000 reference variables and 1000 BankAccount objects.

(E) A single BankAccount object

Question 6 What is the meaning of "method overloading"?

(A) Calling a method too often causes a `MethodOverloadException` exception.

(B) The type of an object changes automatically in an asignment.

(C) Calling a method with too many arguments causes a `MethodOverloadException` exception.

(D) When a method is invoked, the method chosen depends on the referenced object.

(E) When a method is invoked, the version of the method is chosen by its signature.

(F) When an overriden method is invoked, the inherited method with the same signature is executed.

Question 7 One operation that we might want to implement on a Stack and a Queue is "`full()`", which returns `false` if the data structure has room for another item to be added, and `true` otherwise. This operation would be useful

(A) only if the Queue or Stack is implemented using an array.

(B) only if the Queue or Stack is implemented using a linked list.

(C) only for a Queue.

(D) only for a Stack.

(E) never.

Question 8 What is true about the following program?

```
public class F
{
    public static void main(String[] args)
    {
        E e1 = new E();
        E e2 = new E();
        e1.change_u("abc");
        e1.inc_v();
        e2.change_u("xyz");
        e1.change_u("xyz");
        System.out.println( e1.get_u() + e1.get_v() + e2.get_u() + e2.get_v() );
    }
}

class E {
    private String u;
    private float v;
    public E()
    {
        this.u = "";
        this.v = 1.0f;
    }
    public String get_u()
    {
        return u;
    }
    public float get_v()
    {
        return v;
    }
    public void change_u(String u)
    {
        this.u = this.u + u;
    }
    public void inc_v()
    {
        this.v = this.v + 3.0f;
    }
}
```

(A) It will print "`abcxyz4.0xyz1.0`".

(B) It will print "`abc4.0xyz1.0`".

(C) It will print "`xyz4.0xyz1.0`".

(D) It has a syntax error

(E) It produces a run-time error

Question 9 What does the following program print?

```
public class X
{
    public static void main(String[] args)
    {
        Y y1 = new Y(729);
        Y y2 = new Y(1000);
        System.out.println(""+y1+y2);
    }
}

class Y {
    private static int n = 1;
    public Y(int n)
    {
        this.n = n;
    }
    public String toString()
    {
        return ""+this.n;
    }
}
```

(A) 1729

(B) 1000729

(C) 11

(D) 2

(E) 10001000

(F) 7291000

(G) 729729

Question 10 Suppose we have the following definition:

```
class Fruit
{
    public int weight;
    public Fruit() { weight = 0; }
}
```

After the following program fragment is executed, what is the value of `apple.weight`?

```
Fruit apple = new Fruit();
apple.weight = 59;
Fruit stone;
stone = apple;
stone.weight = 37;
```

(A) 59

(B) 37

(C) 0

(D) None of the above, it produces a compile-time error.

(E) None of the above, it produces a run-time error.

Question 11 Consider the following method definitions:

```
public static void a(int j)
{
    j = 3 - j;
}
public static int b(int j)
{
    return 3 - j;
}
public static int test(void)
{
    int i, j;
    i = 1;
    j = 15;
    a(j);
    j = b(i);
    return j;
}
```

If the method `test` is invoked, what are the values of `i` and `j` at the end of `test` (just before it returns)?

(A) `i` is 1, `j` is 1

(B) `i` is 15, `j` is 15

(C) `i` is 1, `j` is 2

(D) `i` is -2, `j` is 12

(E) `i` is 1, `j` is -2

Question 12 Given the following method

```
public static void print(int[] a, int n)
{
    System.out.print("" + a[n] + ", ");
    if (n != 0) print(a, n - 1);
}
```

what will the following fragment print?

```
int[] b = new int[4];
b[0] = 1; b[1] = 2; b[2] = 3; b[3] = 4;
print(b, 3);
```

(A) Nothing, as a recursive function cannot have `void` return type.

(B) 3, 2, 1,

(C) 1, 2, 3,

(D) 1, 2, 3, 4,

(E) 4, 3, 2, 1,

2 General problems

Question 1 For each of the following assertions, state whether it is true or false.

(A) If `first` and `last` are both String variables, then `first.equals(last)` is the same as `(first == last)`.

(B) Any class can implement an interface, but no classes can implement more than a single interface.

(C) If an exception is thrown and is not caught anywhere in the program, then the program terminates.

(D) If class `AParentClass` has a protected instance data `x`, and `AChildClass` is a derived class of `AParentClass`, then `AChildClass` can access `x` but can not redefine `x` to be of a different type.

(E) Assume that `Poodle` is a derived class of `Dog`. Following the declarations `Dog d = new Dog();` and `Poodle p = new Poodle();` the assignment statement `p = d;` is legal even though `p` is not a `Dog`.

(F) We say that Stacks are LILO (or FIFO) and Queues are LIFO (or FILO).

Question 2 What does the following program print?

```
public class P {
    public static void main(String[] args)
    {
        Q a = new Q(6, 1, 8);
        R b = new R(3, 4, 5);
        a.f();
        a.g();
        b.f();
        b.g();
        b.h();
    }
}

class Q {
    private int a;
    protected int b, c;
    public Q(int x, int y, int z) { a = x; b = y; c = z; }
    public void f() { System.out.println("Q.f: " + a + "," + b + "," + c); }
    public void g() { System.out.println("Q.g: " + a + "," + b + "," + c); }
}

class R extends Q {
    private int a, c;
    public R(int x, int y, int z) { super(7,2,9); a = x; b = y; c = z; }
    public void f() { System.out.println("R.f: " + a + "," + b + "," + c); }
    public void h() { System.out.println("R.h: " + a + "," + b + "," + super.c); }
}
```

Question 3 What will be printed by the following program?

```
public class BeerTaster {
    public static void main(String[] args)
    {
        Beer[] pints = new Beer[4];
        pints[0] = new Beer();
        pints[1] = new FinDuMonde();
        pints[2] = new Budweiser();
        pints[3] = new Tartan();
        for (int pint = 0; pint < pints.length; pint++) {
            taste(pints[pint]);
        }
    }
    private static void taste(Beer b)
    {
        b.taste();
    }
}

class Beer {
    public void taste()
    {
        System.out.println("hmmm...");
    }
}

class FinDuMonde extends Beer {
    public void taste()
    {
        System.out.println("Good");
    }
}

class Budweiser extends Beer {
    public void taste()
    {
        System.out.println("Bad");
    }
}

class Tartan extends Beer {
    public void drink()
    {
        System.out.println("Good");
    }
}
```

Question 4 What will this program print?

```
public class Sleep {
    public static void main(String[] args)
    {
        try {
            for (int i = 0; i < 10; i++) {
                Sheep s = new Sheep();
                s.jump();
            }
            System.out.println("zzz...");
        }
        catch (LoudSound s) {
            System.out.println(s);
        }
    }
}

class Sheep {
    private int number;
    private static int total = 0;
    public Sheep()
    {
        this.total++;
        this.number = this.total;
    }
    public void jump() throws LoudSound
    {
        System.out.println("Sheep #"+this.number+" jumped");
        if (this.number == 3)
            throw (new LoudSound(this.number,this.total));
    }
}

class LoudSound extends Throwable {
    int n,t;
    public LoudSound(int n, int t)
    {
        this.n = n;
        this.t = t;
    }
    public String toString()
    {
        return "I was on #"+this.n+" out of "+this.t;
    }
}
```

Question 5 Reformulate the following method without using recursion (You may use loops, conditionals, and a 2 dimensional array instead, to store the values of the arguments as they change):

```
public int mymethod(int a, int b)
{
   if (a <= 0) return 2;
   else if (b <= 0) return 3 * a;
   else return (mymethod(a - 1, b) + 5 * mymethod(a, b - 1));
}
```

3 Programming

Note: For the problems in this section you can use only the methods which appear on page 2, or define your own.

Question 1 Write a recursive method which receives a string as input and returns the reverse of the string (i.e. the string backwards.)

Question 2 Write a method to check whether a given string is a palindrome, using a stack. Assume that the `Stack` class has the methods `top()`, `pop()` and `push()`, with the signatures

```
public Object top()
public Object pop()   // returning the object popped
public void push(Object o)
```

Question 3 Your company is currently working on a project for a personal financial management system. You have been asked to design and implement a class Mortage to represent a home loan. A mortage has a balance (the loan amount,) and a fixed monthly interest rate. It should provide the functionality to: 1) Find out the current outstanding balance, 2) Make a payment, reducing the outstanding balance, 3) Add one month's worth of interest, increasing the outstanding balance. The class should not be responsible for the user interface.

Question 4 Suppose that a building is made up of up to 100 *blocks*, where each block is either a *cube* or a *pyramid*. Write a class to represent buildings, which includes a method to add a new block to the building, and method to compute the total volume of the building, where the total volume is the sum of the volumes of all the blocks in the building. Each block, regardless of its type, has a volume, and it should be possible to obtain the volume of a given block. A pyramid is a block that has a square base and four equilateral triangles on the sides. For this application, it is not necessary to use all this information to represent a pyramid, as a pyramid can be described by the length of a side of the base (b) and its height (h). Then its volume is $\frac{1}{3}b^2h$. A cube is a block which has six squares for sides. It can be described by the length of one of the sides of the squares (l). Its volume is l^3. The code must be object-oriented and concise (Hint: You'll need four small classes.)