University of Bath

**DEPARTMENT OF COMPUTER SCIENCE**
**EXAMINATION**

**CM10134: PROGRAMMING 1**

Day, Day Month Year, fromh:fromm–toh:tom

No calculators may be brought in and used.

Full marks will be given for correct answers to THREE questions.
Only the best three answers will contribute towards the assessment.

1. (a) Is the following statement true or false? Briefly explain your answer.

A && B is the same as B && A for any Boolean expressions A and B.

[6]

(b) Write a program that reads in a series of floating-point numbers and prints out the minimum, maximum and average values. Assume use of the ConsoleReader class. [7]

(c) Write a method that computes the alternating sum of all elements in an integer array. For example, if the method is invoked with the array containing 3 1 4 2 5 9 2 6 3 then it returns $3 + 14 + 25 + 92 + 63 = 7$. [7]

2. (a) Write a simple Bank Account class that rejects negative amounts in the deposit and withdraw methods and rejects withdrawals that would result in an overdraft. [7]

(b) What is wrong with the following loop? Explain 2 ways of fixing the error.

```
int[] v = new int [10];
for (int i = 1; i <= 10; i++)
v[i] = i*i;
```

[6]

(c) For each of the following, say if it is true or false.

(a) Array subscripts must be integers
(b) Arrays cannot contain Strings as elements
(c) Arrays cannot use Strings as subscripts
(d) Parallel arrays must have equal length
(e) Two-dimensional arrays always have the same number of rows and columns
(f) Two parallel arrays can be replaced by one two-dimensional array
(g) Elements of different columns in a two-dimensional array can have different types

[7]

3. (a) Explain the difference between class and instance methods/members. How can this be expressed in your code? [6]

   (b) Can class methods/members be used in instance methods? Explain why (not). [2]

   (c) Can class instance methods/members be used in class methods? Explain why (not). [2]

   (d) In software engineering, solutions to common design problems are often called design patterns. One of them is called Singleton and it describes how one can make sure that a certain class can have only one instance during a program's life-cycle. How would you implement such a Singleton class in Java? [8]

   (e) How would you check that you can indeed have only one instance? [2]

4. (a) Explain the following lines of code (you can assume they are correct):

   (i) public static final double PI = 3.14;

   (ii) protected final void printString(char a)

   (iii) private BankAccount()

   [3]

   (b) Describe the mechanism of error/exception handling in java. [7]

   (c) Why would one use error handling in a constructor? What will happen with the (partial) object? [4]

   (d) (i) Explain the differences between call-by-reference and call-by-value parameter passing?

   (ii) Which systems are used in Java?

   [6]

5. (a) Consider the following four classes. By accident 15 errors slipped in. Detect them and explain why they are errors.

class Run

```
1. import java.util.*;
2.
3. public class Run
4. {
5.
6.    Vector elements;
7.
8.    public Run(int in)
9.    {
10. int t = a;
11. elements = new Vector();
12. for(int t = 0; t <in; t++)
13.     {
14. if(t%2==0)
15.     elements.addElement(new ElementA("a"));
16. else
17.     elements.addElement(new ElementB(t));
18.
19.     }
20.    }
21.
22.    public String toString()
23.    {
24. String res = ";
25. int s = elements.size();
26. for(int i=0; i<s; i++)
27.     {
28. res += elements.elementAt(i);
29. res += "\n";
30.     }
31. return res;
32.    }
33.
34.    public Vector giveElements()
35.    {
36. return elements;
37.    }
38.
```



*Question 5 continues on next page ...*

*Question 5 continued ...*

```
39.    public static void main(String[] argv)
40.    {
41. Element a = new Element();
42. Run r = new Run(4);
43. System.out.println(r);
44. ElementA b = new Element(5);
45. Vector el = r.giveElements();
46. System.out.println((ElementA) el.elementAt(1));
47.
48.    }
49.
50.}
```

class Element

```
1.// abstract class Element
2.
3.public abstract class Element
4.{
5.
6.
7.    public Element()
8.    {
9. System.out.println("Element created");
10.    }
11.
12.    protected abstract String printContents();
13.
14.
15.    protected String printType()
16.    {
17. return "Element "
18.    }
19.
20.    public String toString()
21.    {
22. printType();
23.    }
24.
25.}
```



*Question 5 continues on next page ...*

*Question 5 continued ...*

class ElementA

```
1.// public class ElementA
2.
3.public class ElementA extend Element
4.{
5.    String el;
6.
7.    public ElementA()
8.    {
9. System.out.println("ElementA created");
10. el = "nothing";
11.    }
12.
13.    public ElementA(String input)
14.    {
15. System.out.println("ElementA created with input");
16. el = input;
17.    }
18.
19.    /* protected constructor */
20.    protected ElementA(a)
21.    {
22. System.out.println("ElementA created with int");
23. el = Integer.toString(a);
24.    }
25.
26.    protected String printContents()
27.    {
28. return el;
29.    }
30.
31.    protected String printType()
32.    {
33. return "ElementA inherits from "  super.printType();
34.    }
35.
36.    public String toString()
37.    {
38. return printType() +
39.     "and contains " + printContents();
40.
41.
42.}
```



*Question 5 continues on next page ...*

*Question 5 continued ...*

class ElementB

```
1./ public class ElementB
2.
3.
4.
5.public class ElementB extends ElementA
6.{
7.
8.    Vector in;
9.
10.    public ElementB()
11.    {
12. System.out.println("ElementB created");
13. in = new Vector();
14.    }
15.
16.
17.    public ElementB(String el)
18.    {
19 System.out.println("ElementB created with input");
20. super(el);
21.
22.    }
23.
24.    public ElementB(int a)
25.    {
26. super(a);
27. System.out.print("ElementB created with int");
28. in = new Vector();
29. final int el = 2;
30. System.out.println("and el equals to " + this.el);
31. el++;
32. for(int i=0;i<a; i++)
33.     {
34. in.addElement(new Integer(i + el));
35.     }
36.    }
37.
```

*Question 5 continues on next page ...*

*Question 5 continued ...*

```
38.    protected String printContents()
39.    {
40. return el + " " + in.toString();
41.    }
42.
43.    protected String printType()
44.    {
45. return "ElementB inherits from " + super.printType();
46.    }
47.
48.    public String toString()
49.    {
50. return printType() +
51.     "and contains " + printContents();
52.    }
53.
```

[15]

(b) Predict the output of this program. [5]