

CSIS 0911B Computer Concepts and Programming
for non-Engineering students (Java)

Assignment 4

Assigned 2:00pm Nov 20, 2000; dead 2:00pm Dec 4, 2000.

Assignment box: R5, lift lobby, 3/F Chow Yei Ching building

Submission: Write down your answers to the questions. Save your program in section 3 in a floppy diskette. Also print a copy of your program. Put all the above into an envelope and put it into assignment box R5. **Remember to write down your name, university ID and your curriculum** on the envelope. Send a mail to the tutor (ydsun@csis.hku.hk) indicating that you have submitted an assignment.

Section 1: Short Questions

1. Multiple multiple choice: Consider the following program. At the position marked as “// ??”, what is the value of arr1, arr2, arr3 and arr4? Choose one below for each of them.

```
public class InitArray {
    static double[] arr1;
    static double[] arr2 = null;
    public static void main(String[] args) {
        double[] arr3, arr4 = new double[0];
        // ??
    }
}
```

- (a) It contains the value null.
 - (b) It points to an array of size 0.
 - (c) It is undefined, and using the value will cause a run-time error.
 - (d) It is undefined, and using the value will cause a syntax error.
 - (e) It is undefined because the program does not compile.
2. Multiple multi-multiple choice: Which of the following statements are true for each of the following position? Choose all that apply for each position.

```
public class ArrCompare {
    public static void main(String[] args) {
        int [] arr1 = { 2, 3, 4 };
        int [] arr2 = { 2, 3, 4 };
        int [] arr3 = arr1;
        // (1) What happens here?
        arr1[2] = 5;
        // (2) What happens here now?
    }
}
```

- (a) arr1.equals(arr2) would return true.
 - (b) arr1.equals(arr3) would return true.
 - (c) arr1 == arr2 would return true.
 - (d) arr1 == arr3 would return true.
3. Draw a diagram showing how the memory would look like for arr when the following program is executed.

```

public class Prog {
    public static void main(String[] args) {
        String[][] arr = {
            { "Hello", "World" },
            { "Foo", "Bar", "FooBar" }
        };
        // What does arr looks like here?
    }
}

```

4. Draw a diagram showing how the memory would look like when the program is executed to the position marked “// ??” in the following program.

```

public class TestFill {
    public static void main(String[] args) {
        int[] arr = { 2, 3 };
        int[][] matrix = new int[3][2];
        java.util.Arrays.fill(matrix, arr);
        // ??
    }
}

```

5. In Java, every object is created by `new` either directly or indirectly through a method. That is, except one special class of object, which has special language support. State the name of this special class of objects and the ways to create an object of that class other than calling `new` or a method.
6. Write a method that takes two `int` type values as input, and return both the quotient and the remainder them. Define all `class`'s that you use to hold the return values.
7. Suppose we want to represent a planet in the solar system as an object of class `Planet`. In the object, it contains two fields: one called `name` is a `String` containing its name, and one called `satellites` is an array of `String` containing the names of all its satellites. Define the class, with a constructor taking a `String` and an array of `String` as argument for initializing the object.
8. For the last question, write a method `print()` in the class to print out the name of the planet and all the satellites. For example, if we have

```

String[] earth_sat = { "Moon" };
Planet earth = new Planet("Earth", earth_sat);

```

Then we want `earth.print()`; to print out the following:

```

Earth has 1 satellite(s):
Moon.

```

9. It turns out that a class reference of a particular type can exist within the class definition of itself. This mechanism is usually used to build more complicated data structures like a linked list. Consider the following class:

```

class List {
    int data;
    List next;
    public static void main(String[] args) {
        List a = new List();
        List b = new List();
        List c = new List();
        a.data = 10;
    }
}

```

```

        b.data = 7;
        c.data = 15;
        a.next = b;
        b.next = c;
        // (a)
    }
}

```

Draw a picture to show what the memory contains when the program is at (a) marked in the comment of the class.

10. In the last question, what is meant by the statement `b.next.data = 8;` if it is written at the position (a) marked in the program? Write a statement to do the same thing using only a (i.e., don't use b and c).

Section 2: Program Debugging

The following program is an attempt to modify the calendar printing program discussed during the lecture, so that it prints out a yearly calendar instead of a monthly calendar. To do that, the program defines a class that represents a monthly calendar, supporting two methods—to print a week of the calendar, and to check whether the calendar is printed completely. Unluckily, the program contains a number of errors. Fix all errors in the program, so that it prints out a yearly calendar.

```

import chapman.io.*;
import java.util.*;

// Hold a partially-printed calendar
class MonthCalendar {
    // Create a calendar, initialize it so that the next time printWeek is
    // called, the first week is printed
    public void MonthCalendar(int y, int m) {
        month = m;
        day1 = new GregorianCalendar(y, m, 1);
        next_to_print = new GregorianCalendar(y, m, 1);
        while (next_to_print.get(Calendar.DAY_OF_WEEK) != Calendar.SUNDAY)
            next_to_print.add(Calendar.DATE, -1);
    }
    // Return true if and only if something is still to be printed
    public static boolean notDone() {
        return next_to_print.before(day1) ||
            next_to_print.get(Calendar.MONTH) == month;
    }
    // Print a week
    public void printWeek() {
        for (int i = 0; i < 7; ++i) {
            if (next_to_print.before(day1) ||
                next_to_print.get(Calendar.MONTH) != month)
                System.out.print("  ");
            else
                Fmt.printf("%3d",
                    next_to_print.get(Calendar.DAY_OF_MONTH));
            next_to_print.add(Calendar.DATE, 1);
        }
    }

    private GregorianCalendar next_to_print; // The last day printed
    private GregorianCalendar day1;        // The first day of week
    private static int month;
}

```

```

// Print the Year calendar for one year
public class YearCal {
    public static void main(String[] args) {
        StdIn in = new StdIn();
        System.out.print("Year? ");
        int year = in.readInt();
        // Each time, print 3 months
        for (int i = 0; i < 12; i+=3) {
            MonthCalendar c1, c2, c3;
            c1 = c2 = c3 = new MonthCalendar(year, i);
            while (c1.notDone() && c2.notDone() && c3.notDone()) {
                c1.printWeek();
                System.out.print("      ");
                c2.printWeek();
                System.out.print("      ");
                c3.printWeek();
                System.out.println();
            }
            System.out.println();
        }
    }
}

```

After fixing the whole program, re-introduce each error into the program one-by-one, and compile the program each time an error is re-introduced. Write down the error message that the compiler outputs, and the intuitive meaning of the error message.

Section 3: Program Development

When creating an array, we need to know the exact size of the array when the array is created. There is no way to expand or contract an array afterwards. There are many applications that want a “resizable” array, because the exact number of elements is not known when the array is created.

This can be achieved in Java by reallocating an array whenever the original array is not sufficient to hold the array. When this happens, we copy all the elements from the old array to the new one.

Write a Java class for representing a resizable array of integers. The class, called `IntArray`, should have the following constructors and methods. Here `ia` denotes a reference to `IntArray`.

- Constructor: `IntArray(initial_size)`, where `initial_size` is an integer specifying the initial size of the array. All elements are initially 0.
- Read: `intAt(index)`: return the integer stored in the array at the integer index `index`. A run-time error should be generated if `index` is out of range.
- Write: `storeAt(index, value)`: store an integer value into the array at the integer index `index`. Return nothing. A run-time error should be generated if `index` is out of range.
- Size: `getSize()`: return the current size of the array.
- Resize: `resize(size)`: change the capacity of the array to an integer `size`. If `size` is different from the current size of the array, a new array should be allocated, and the old content of the array should be copied to the new array. If `size` is smaller than the original size, the extra elements in the original array are lost. If `size` is larger than the original size, the extra elements in the new array takes the value 0. Return nothing.
- Write and resize: `storeAndResize(index, value)`: store an integer value at an integer index `index`. If `index` is larger than or equal to the current size of the array, it also expand the array. If this happens, the array is expanded to either a size of `index+1`, or double the original size, whichever the larger. A run-time error should be generated if `index` is negative. Return nothing.

With this class, write a program that reads integers until it reads -1, and prints out the median of the integers read (not counting -1). For example:

```
Type the numbers, terminated by -1:
```

```
2
```

```
9
```

```
8
```

```
3
```

```
7
```

```
6
```

```
-1
```

```
The median is 6.5.
```