Question 1

a) Answer the questions below:

[3 marks]

i) Describe the difference between a Java *object* and a Java *class*.

(1 mark)

ii) Describe the concepts of *pass-by-reference* and *pass-by-value*.

(2 marks)

	Do not write in this column	
i)	**1/3**	
ii)	**2/3**	
		3 marks

b) Consider the 3 Java classes, **`Animal`**, **`Cat`** and **`Dog`** with the following requirements:

- **`Cat`** and **`Dog`** are both a subclass of **`Animal`**.
- All kinds of **`Animal`** require attributes of an **`owner`** (a **`String`**) and an **`age`** (an **`int`**).
- Objects of type **`Cat`** require a method called **`meow()`** to allow them to *meow* (i.e. the noise that a cat makes). This method should print out a message with the text `"meow"`.

[14 marks]

i) Draw a simple UML diagram to represent the 3 Java classes above and the relationships between them. You only need to show the class names, i.e. you do not need to show any fields or methods.

(1 mark)

ii) Write example code for the 3 Java classes **`Animal`**, **`Cat`** and **`Dog`**. Design your classes to minimise the number of field definitions, by utilising *inheritance*. Provide *getter* and *setter* methods for all attributes following the usual naming conventions for such methods. **`Cat`** and **`Dog`** should both implement a constructor allowing the **`owner`** and **`age`** variables to be initialised.

(13 marks)

	Do not write in this column
i)	**1/14**
ii)	**13/14**
	14 marks

c) Write a Java program that takes a **`String`** from the command line and then writes it back to the screen replacing any spaces with a dash (**`-`**). Your program must contain a method that replaces any spaces from the input **`String`**; this method's signature should be:

```
public String replaceSpaces(String str)
```

Figure 1 shows a typical use and output of the program described above.

```
C:\> java SpacesReplacer "What a lovely day."
Replace spaces: What-a-lovely-day.
```

Figure 1

Hint: You can use the method described in **Figure 2**.

```
char charAt(int index)
        Returns the char value at the specified index.
```

Figure 2

[8 marks]

	Do not write in this column
	8 marks

Question marking: $\frac{\quad}{3}+\frac{\quad}{14}+\frac{\quad}{8}=\frac{\quad}{25}$

Question 2

a) Consider the Java method shown in **Figure 3** and answer the questions below:

```
public void mystery(ArrayList<Integer> list) {
  for (int i=1; i < list.size(); i++) {
    if (list.get(i-1) < list.get(i)) {
      list.remove(i);
      list.add(i, list.get(i-1));
    }
  }
}
```

Figure 3

[9 marks]

i) Assume the method in **Figure 3** is called with the following **`ArrayList`**: **`[1,2,3,4]`**. Determine what will be the final contents of the **`ArrayList`**, after the method has been called.

(1 mark)

ii) Now assume the method in **Figure 3** is called with the following **`ArrayList`**: **`[5,5,5,0,5,5,5]`**. What will be the final contents of the **`ArrayList`**, after the method has been called?

(1 mark)

iii) Name TWO differences between standard *arrays* and **`ArrayList`**s. Your answer must also include Java code to declare and initialise the set of integer values **`{3,5,7,9}`** as a standard *array* and as an **`ArrayList`**.

(7 marks)

	Do not write in this column
i)	**1/9**
ii)	**1/9**
iii)	**7/9**
Difference 1	
Difference 2	

	Do not write in this column	
Declare and initialise *array* of integer values `{3,5,7,9}`		
Declare and initialise `ArrayList` of integer values `{3,5,7,9}`		
		9 marks

b) Describe TWO different ways to make an *object eligible for Garbage Collection*. Your answer must include ONE Java code illustrative example for each of the described approaches.

[5 marks]

	Do not write in this column	
Approach 1		
Approach 2		
		5 marks

c) The questions below refer to the concept of *inheritance* in Java:

[4 marks]

i) Briefly describe the concept of *inheritance* and state TWO of its benefits.

(2 marks)

ii) Write an example class *declaration* for a class called **`Student`** that inherits from a class called **`Person`**.

(1 mark)

iii) Write the Java code that creates an instance of type **`Student`**, using its *default* constructor.

(1 mark)

	Do not write in this column
i)	**2/4**
ii)	**1/4**
iii)	**1/4**
	4 marks

d) You are required to write the code for a Java method called **`gradeExam`** that converts a student's exam percentage into a grade (e.g. **`A`**, **`B`**, **`C`**); the grade must then be printed to the screen. The method's signature is **`public void gradeExam(int[] marks)`**and the parameter **`marks`** is an array containing the marks of all students in a class (e.g. **`{67`**, **`81`**, **`50`**, **`34}`**). Your code must iterate through the array printing out the grade that each student achieved. The below table describes the exact criteria for each grade.

[7 marks]

Grade	**Exam Percentage Mark**
A	Mark greater or equal to 70
B	Mark between 50 and 69 (inclusive)
C	Mark between 40 and 49 (inclusive)
F	Mark below 40

Table 1

	Do not write in this column
	7 marks

Question marking: $\frac{\quad}{9}+\frac{\quad}{5}+\frac{\quad}{4}+\frac{\quad}{7}=\frac{\quad}{25}$

Question 3

a) The code in **Figure 4** has lines numbered **1-27**; the code does not compile. For each of the compiler errors shown below in **Figures 5–7**, briefly explain the error and write the correct line of code that would fix the error, in the space provided.

Your answer must also indicate the output of the program, after all the compilation errors have been fixed.

[7 marks]

```
 1  public class Interesting {
 2    public int incrementCounter(int i) { return (i + 1); }
 3
 4    public static void main(String[] args) {
 5      final int k = 1;
 6      String[] myStrings = {'first', 'second'};
 7      int[] y;
 8      boolean b = false;
 9
10      Interesting obj = new Interesting();
11
12      if (myStrings.length() == 2) {
13        // Print the second string in myStrings to screen.
14        System.out.println(myStrings[1]);
15      }
16      for (int i = 0; i < 100;) {
17        y[i] = k;
18        i = obj.incrementCounter(i);
19      }
20      if (!b) {
21        System.out.println("This line should always be printed.");
22      }
23      if (b = true) {
24        System.out.println("This line should never be printed.");
25      }
26    }
27  }
```

Figure 4

```
Interesting.java:6: error: unclosed character literal
    String[] myStrings = {'first', 'second'};
                          ^
```

Figure 5

```
Interesting.java:12: error: cannot find symbol
    if (myStrings.length() == 2) {
                 ^
  symbol:   method length()
  location: variable myStrings of type String[]
```

Figure 6

```
Interesting.java:17: error: variable y might not have been initialized
       y[i] = k;
       ^
```

Figure 7

	Do not write in this column
Compiler error – Figure 5	
Error description	
Correct line of code to fix the error	
Compiler error – Figure 6	
Error description	
Correct line of code to fix the error	
Compiler error – Figure 7	
Error description	
Correct line of code to fix the error	
Output of the program	
	7 marks

b) Answer the questions below with reference to the Java code in **Figure 8**:

[8 marks]

```
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  for (int i=10; i <= 110; i = i + 50) {
    for (int j = 10; j <= 110; j = j + 50) {
      g.drawLine(i,10,j,60);
    }
  }
}
```

Figure 8

i) Draw the picture that will be generated by the method in **Figure 8**. Your answer must include an explanation for the picture drawn.

(3.5 marks)

ii) Write a Java program called **`DrawImage`** that displays the picture you identified in *part i)*, using the method in **Figure 8**. It must include a **`main()`** method.

Hint: The complete Java program will require you to write the code for TWO classes.

(4.5 marks)

	Do not write in this column
i)	**3.5/8**
Image that will be displayed	
Explanation	
ii)	**4.5/8**

	Do not write in this column	
		8 marks

c) Describe the concept of *layout manager* and indicate TWO examples of Java *layout managers*.
[4 marks]

	Do not write in this column	
		4 marks

d) The following questions refer to the Java code (with lines numbered **1–14**) in **Figure 9**.

[6 marks]

```
 1   public class TextManipulation {
 2     public static void main(String[] args) {
 3       String book1 = "Towards zero";
 4       StringBuilder book2 = new StringBuilder("Alibi");
 5       int index = book1.indexOf('a');
 6       String author = new String(" by AC");
 7
 8       book2.setCharAt(0, book1.charAt(0));
 9       book2.setCharAt(1, book1.charAt(book1.length()-1));
10       book2.insert(1, book1.charAt(4));
11       book2.insert(3, (book1.substring(index, index+2) + " "));
12       System.out.println(book2);
13     }
14   }
```

Figure 9

i) Determine the *initial capacity* of the variable **`book1`** in **Figure 9**. Justify your answer.

(2 marks)

ii) Indicate the value of variable **`book2`**, after the code in each of the lines **`8-11`** is executed.

(2 marks)

iii) Give ONE way in which you can *concatenate* the two *strings* shown in *lines* **`3`** and **`6`** of **Figure 9** and indicate the value of the resulting *string*. You must write the Java code to do this.

(2 marks)

	Do not write in this column
i)	**2/6**
ii)	**2/6**
`line  8`:	
`line  9`:	
`line 10`:	
`line 11`:	

	Do not write in this column
iii)	**2/6**
	6 marks

Question marking: $\frac{}{7}+\frac{}{8}+\frac{}{4}+\frac{}{6}=\frac{}{25}$

Question 4

a) The questions below refer to *exceptions* and *assertions*:

[7 marks]

i) Consider the Java program in **Figure 10** (with lines numbered **1–15**), which compiles and runs successfully. Determine the output of the program, when *assertions* are disabled <u>and</u> when *assertions* are enabled. Justify your answers.

(4 marks)

```
 1   public class Example {
 2     public static void main(String[] args) {
 3       try {
 4         // String str = "";
 5         // char c = str.charAt(1);
 6         assert(false);
 7       }
 8       catch (Exception ex) {
 9         System.out.println("Caught an exception!");
10       }
11       catch (Error err) {
12         System.out.println("Caught an error!");
13       }
14     }
15   }
```

Figure 10

ii) Modify the code in **Figure 10** such that lines **4-5** are no longer Java comments, and delete the code in line **6**. The resulting program will still compile and run successfully. Determine its output <u>and</u> justify your answer.

(3 marks)

	Do not write in this column
i)	**4/7**

	Do not write in this column
ii)	3/7
	7 marks

b) Consider the Java statement in **Figure 11** <u>and</u> answer the questions below:

```
int[][] table = new int[5][5];
```

Figure 11

[4 marks]

i) Write a Java statement that stores the value **`4`** in the second row of the third column of the array defined in **Figure 11**.

(1 mark)

i) Write Java code to store the numbers **`1-5`** in the first row, numbers **`6-10`** in the second row, and so on.

Note: You are <u>not</u> required to write a complete Java program.

(3 marks)

	Do not write in this column
i)	1/4
ii)	3/4
	4 marks

c) The questions below refer to File I/O in Java:

[9 marks]

i) Java I/O is usually defined using *streams*. Briefly describe the concept of *streams* in Java.

(1.5 marks)

ii) Write a Java program called **`StudentMarksWriter`** that creates a file named **`marks.dat`** and writes a list of student marks (e.g. **`100`**, **`45`**, **`83`**, **`42`**) into the file. Each mark should be written to the file on a separate line. You can assume that all marks are available in an array of integer values (you should declare this array in your own code and add some example marks into it). The file should be created in the same directory as the program. If the file already exists, then the program should display the message **`"File already exists"`** and then terminate the program.

(7.5 marks)

	Do not write in this column	
i)	**1.5/9**	
ii)	**7.5/9**	
		9 marks

d) Write a block of Java code that creates a **`JButton`** and places it inside of a **`JFrame`**. The **`JButton`** should contain the text "**`Click Me`**". Add appropriate *event handing* code so that the text changes to "**`I have been clicked`**" after somebody clicks the button.

Note: You are <u>not</u> required to write a complete Java program.

[5 marks]

	Do not write in this column	
		5 marks

Question marking: $\frac{\quad}{7}+\frac{\quad}{4}+\frac{\quad}{9}+\frac{\quad}{5}=\frac{\quad}{25}$