Revision Lecture

Topics:

- EBU4201 course: brief outline of topics
- Exam format & Study hints
- Sample exam questions & How to answer them
- Final advice

Course Topics: Outline (1/4)

- **Java Basics**
  - Variables // Basic/primitive types // Reserved Words // Operators
  - Control Structures // Constructors // Type Casts // Arrays
- **Object Oriented Concepts & Programming**
  - Classes and Objects // Abstraction and Inheritance
  - Initialisation and Constructors // Methods
  - Method Overloading // Data Hiding/Encapsulation
  - Access Control Levels // Accessor and Mutator Methods
  - Subclasses (**is-a**) and Aggregation (**has-a**) // Overriding methods
  - `final` and `static`: Variables, Methods and Classes
  - Abstract classes and Interfaces

Course Topics: Outline (2/4)

- **GUI**
  - Making GUIs (`JFrame`, `JButton` classes in `javax.swing`)
  - Event Handling: Events, Sources & Listeners (`java.awt.event.*`)
  - Graphics Classes: `Color`, `Font`, `FontMetrics`, `JLabel`
  - Layout Managers: `FlowLayout`, `GridLayout`, `BorderLayout`
- **Garbage Collection**
  - Heap and Stack // Overloaded Constructors, `this()` and `super()`
  - Life of Objects // Scope of a Variable // `null` References
- **Strings**
  - Checked and unchecked exceptions // `try`/`catch`/`finally` blocks
  - String classes: `String`, `StringBuffer`, `StringBuilder`, `Character` and `Scanner`

Course Topics: Outline (3/4)

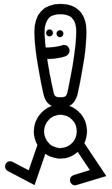
- **Numbers**
  - `Math` Class and methods // Wrapper classes and Autoboxing
  - Formatting: Numbers and Dates // Recursion
- **Exception Handling**
  - Checked and unchecked exceptions // `try`/`catch`/`finally` blocks
  - `throw` *versus* `throws` // Declaring and Catching exceptions
- **File I/O**
  - Saving Data // Java I/O Streams
  - Reading from/Writing to a Text File: `FileReader`, `FileWriter`, `BufferedReader`, and `BufferedWriter` classes
  - Exceptions from File I/O Classes // File objects (`java.io.File`)

Course Topics: Outline (4/4)

- **Collections**
  - `java.util.*` // `ArrayList` // `ArrayList<type>`
  - `Iterator` // Sorting algorithms: Bubble sort, Selection sort
  - `Comparable` interface
- **Classpath, Packages, JARs**

Exam Format

- Format:
  - Need to answer all 4 questions.
  - Duration of exam: 2 hours.
  - Before starting to answer any particular question, make sure you read (or at least skim through) all the questions first.



It is very important to revise and be familiar with all the topics covered in the course!

- Most questions have some amount of programming!
  - You are expected to be able to program simple methods and/or classes in Java during the exam.
- In general, you should be able to find a skeletal solution to the questions directly from your lecture notes!

Study Hints

- Make sure you understand the Lab Exercises and what they are trying to teach you.
  - Lab Exercises often focus on specific skills e.g.:
    - concept of an object // abstract/concrete classes
    - data encapsulation // string manipulation
    - how to declare/catch exceptions // how to write a GUI
- Typical examples of questions:
  - Write a Java method and/or class that does **X**.
  - Define "Some Java or OO Concept – e.g. method overloading, inheritance, keywords". Give an example.
  - Given the following code … What does it do?
  - Explain how you would do **X** in Java?

Sample Questions & Answers (1/3)

- Determine whether the following class is correct or not. *If it is in error*, specify what is wrong with it. *If it is correct*, describe what the program does.

```
public class Norm {

  // define instance data
  private double x, y; // x,y position of the point

  // define constructor
  public void Norm(double x, double y) {
     this.x = x;  this.y = y;
  }

  public void calcNorm()   {
     return Math.sqrt(x*x+y*y);
  }


  public static void main(String[] args) {
    x=3;  y=4;
    System.out.println(" The norm is " + Norm.calcNorm());
  }
}
```

(*Guideline answer*) Sample Q&A (1/3) – 1/2

- The class is **incorrect** because,
  - The constructor is wrongly declared. Because it has a `void` return type, this means that `Norm()` isn't a constructor but a method to set the values of `x` and `y` to an object.
  - The return type of `calcNorm()` is `void` so it cannot return a value as indicated. To fix this, the method's return type should be `double`.
  - The program attempts to initialise variables `x` and `y` in `main()` before declaring them. These variables should be declared as `double`.
  - Using `Norm.calcNorm()` in the `main()` function to call method `calcNorm()` is wrong, since by this point, no instance of the `Norm` class has yet been created. In order to call the `calcNorm()`, an instance of class `Norm` needs to be created first.

(*Guideline answer*) Sample Q&A (1/3) – 2/2

- The correct code is,

```
public class Norm {
  // define instance data
  private double x, y; // x,y position of the point

  // define constructor
  public Norm(double x, double y) {
    this.x = x;   this.y = y;
  }
  public double calcNorm() {
    return Math.sqrt(x*x+y*y);
  }
  public static void main(String[] args) {
    double x, y;  x=3;   y=4;
    Norm norma = new Norm(x,y);
    System.out.println(" The norm is " +
                                 norma.calcNorm());
  }
}
```

Sample Questions & Answers (2/3)

- Create two new classes called **NewMail** and **NoMail**, both as subclasses of the class **Message** (defined below). Demonstrate how they can be instantiated and displayed. They should display the messages **"You have new email!"** and **"You have no new email!"**, respectively, together with the message contained in the variable **news**.
  - **Note**: You need to write a class for each message that inherits from **Message**.

```
public abstract class Message {
  void display(String news);
}
```

(*Guideline answer*) Sample Q&A (2/3)

- Classes **`NewMail`** and **`NoMail`** are implemented as,

```
public class NewMail extends Message {
  public void display(String news) {
    System.out.println("You have new email!"+ news);
  }
}
public class NoMail extends Message {
  public void display(String news) {
    System.out.println("You have no new email!" +
                                        news);
  }
}
```

- The classes can be instantiated & displayed by using a test program,

```
public class TestMessage {
  public static void main(String args[]) {
    Message greatNews = new NewMail();
    Message sosoNews = new NoMail();
    greatNews.display("You won!");
    sosoNews.display("Try later!"); }
}
```

"Demonstrate how they can be instantiated and displayed"

Sample Questions & Answers (3/3)

- Do you need to *deallocate memory* when an object is no longer in use? If yes, how? If not, why?
- What is a *class variable*? When would you use it?
- What is the difference between a *concrete class* and an *abstract class*?

(*Guideline answers*) Sample Q&A (3/3)

- It is not necessary to do this, because Java has Garbage Collection which collects objects that are no longer in use, and destroys them. This means that unused memory is automatically released.

- A type of variable that is created when its class is created, rather than when an instance of the class is created. There may be *n* instances of the class, but there will only be one instance of a class variable. Class variables are also referred to as static variables, and the modifier **`static`** is used to declare them.
- Example of when to use a class variable: when we want to count how many instances of a class are made.

- An abstract class cannot be instantiated, whereas a concrete class can be instantiated.
- Abstract classes are usually extended by one or more concrete classes.

Past Exam Questions & Marking Criteria (1/2)

a) Write a Java program called **`StringTest`** that takes a **`String`** from the command line argument, and does the following:

- changes all letters to lowercase;
- replaces any spaces with a comma (`,`);
- prints out the **`String`** in reversed order.

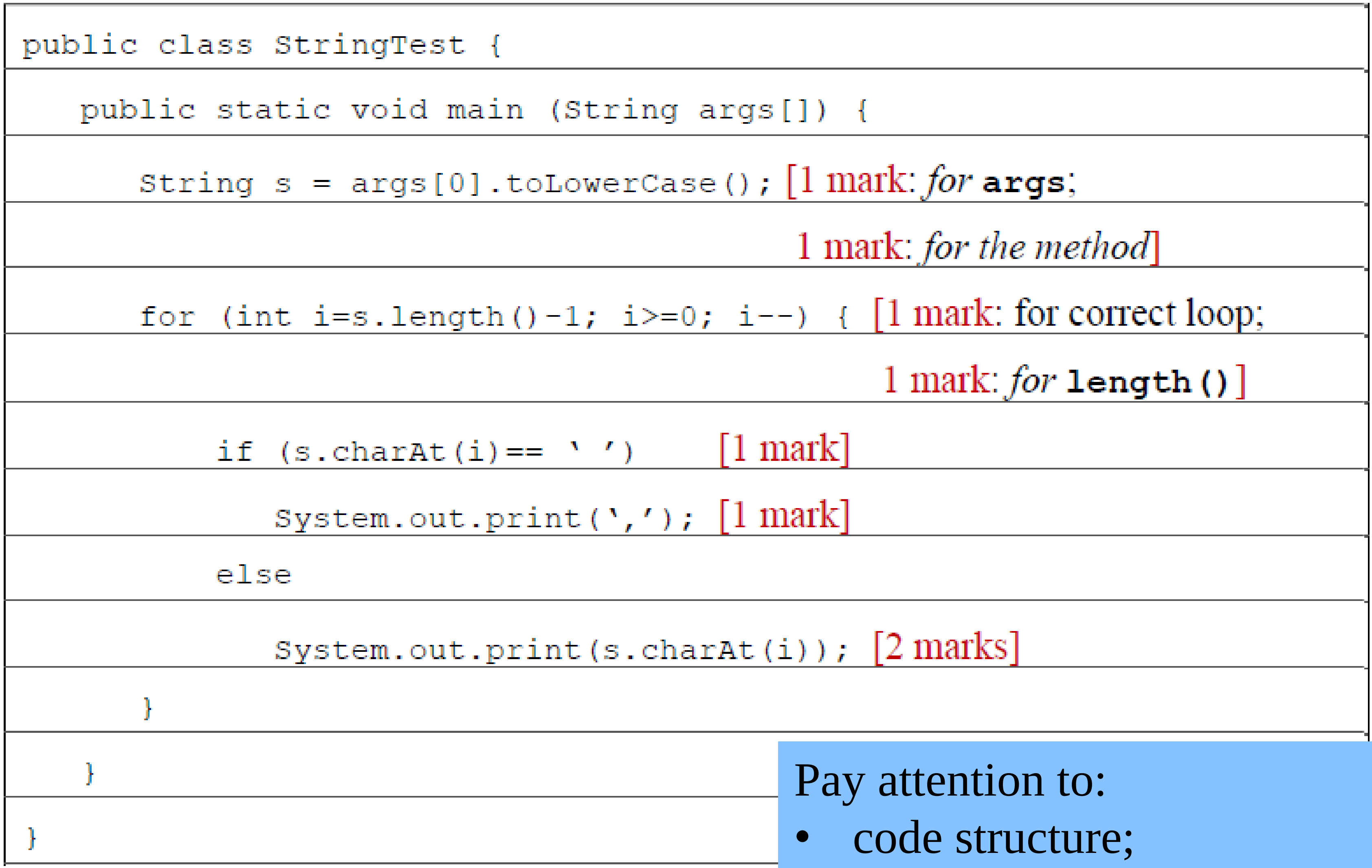
Figure 7 gives an example of running this Java program:

[8 marks]

```
> java StringTest "EBU4201 Java Programming"
gnimmargorp,avaj,1024ube
```

Figure 7

(*Guideline answer*) Past Exam Questions & Marking Criteria (1/2)



```
public class StringTest {
   public static void main (String args[]) {
      String s = args[0].toLowerCase(); [1 mark: for args;
                                         1 mark: for the method]
      for (int i=s.length()-1; i>=0; i--) { [1 mark: for correct loop;
                                              1 mark: for length()]
         if (s.charAt(i)== ' ')      [1 mark]
            System.out.print(','); [1 mark]
         else
            System.out.print(s.charAt(i)); [2 marks]
      }
   }
}
```

Pay attention to:

- code structure;
- question's requirements

Past Exam Questions & Marking Criteria (2/2)

c) Assuming **`Student`** is a concrete class, consider the code fragment in **Figure 8** (with lines numbered **`1–5`**) and answer the following questions:

[4 marks]

```
1    Student s1 = new Student();
2    Student s2 = new Student();
3    Student s3 = s1;
4    s1 = s2;
5    s3 = s2;
```

Figure 8

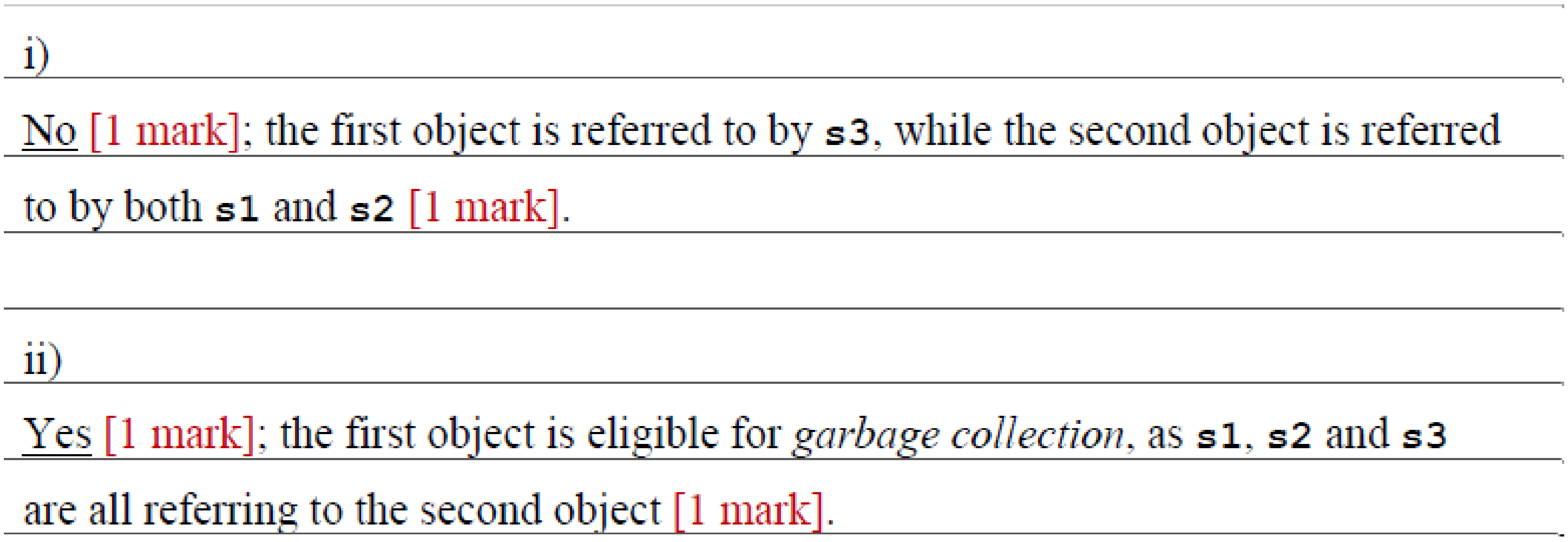
i) After the execution of the statement in line 4, are there any objects eligible for *garbage collection*? Explain.

(2 marks)

ii) After the execution of the statement in line 5, are there any objects eligible for *garbage collection*? Explain.

(2 marks)

(*Guideline answer*) Past Exam Questions & Marking Criteria (2/2)



i)

No [1 mark]; the first object is referred to by **`s3`**, while the second object is referred to by both **`s1`** and **`s2`** [1 mark].

ii)

Yes [1 mark]; the first object is eligible for *garbage collection*, as **`s1`**, **`s2`** and **`s3`** are all referring to the second object [1 mark].

Don't forget to justify your answers.

And finally …

- When revising:
  - **Do the old exams**! Answer them once open book and then see if you can do it without the book!
  - Think of areas that may not be covered by past exams, as these could still be covered.
    - **Examples**:
      - Was there a question about inheritance?
      - What about abstract classes or interfaces?
      - What about recursion?
  - **Review your coursework**: a basic subset of knowledge questions is very probable.

Good Luck!! ☺