**44-542 Object Oriented Programming Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Exam 02 (100 points) Part 2 KEY** *please print*

1. (8 pts) Suppose we have a **Course** class that implements **Comparable<Course>**, with the **compareTo** method comparing courses on the basis of **courseNumber**. In method **main** we want to sort courses alphabetically by **courseTitle**. Complete the statement below so that courses will be sorted by **courseTitle**.

You may assume **courseTitle** is of type **String**. You may also assume the **Course** class provides a getter method for **courseTitle**.

Your code should fit in the space below. Do not write on the back.

**Collections.sort(myCourses, new Comparator<Course>()**

**{**

**@Override**

**public int compare(Course course01, Course course02)**

**{**

**return course01.getCourseTitle().**

**compareTo(course02.getCourseTitle());**

**}**

**});**

1. (8 pts) Consider the code for class City below.

**public class City**

**{**

**private String cityName;**

**private ArrayList<Integer> rainAmt;**

**public City (String cityName)**

**{**

**this.cityName = cityName;**

**rainAmt = new ArrayList<Integer>();**

**}**

**public void addAmt (int amt)**

**{**

**rainAmt.add(amt);**

**}**

**}**

In the space below, rewrite the code for **addAmt** so that an **IllegalArgumentException** is thrown, with message **Invalid precipitation amount**, if the amount to be added is less than 0 or greater than 100.

**public void addAmt (int amt)**

**{**

**if(amt < 0 || amt > 100)**

**{**

**throw new IllegalArgumentException("Invalid precipitation amount");**

**}**

**rainAmt.add(amt);**

**}**

1. (6 pts) Make the necessary changes to the class Courses below, by adding the code for the iterator method and adding any other necessary code so that the result will compile and run.

***You do not have to include package or import statements.***

***Do not add any methods or private instance variables.***

***Your code should fit in the space provided. Do not write on the back.***

**public class Courses implements Iterable<Course>**

**{**

**private ArrayList<Course> courses;**

**public Courses()**

**{**

**courses = new ArrayList<Course>();**

**}**

**public void addCourse(Course course)**

**{**

**courses.add(course);**

**}**

**@Override**

**public Iterator<Course> iterator()**

**{**

**return courses.iterator();**

**}**

**}**

1. (6 pts) Assume method **mystery** is defined as shown here:

**public static String mystery(String str1, String str2)**

**{**

**if(!str1.contains(",") && (!str2.contains(",")))**

**{**

**return "done";**

**} else**

**{**

**if(str1.indexOf(",") > 0)**

**{**

**str1 = str1.substring(str1.indexOf(",") + 1);**

**}**

**if(str2.indexOf(",") > 0)**

**{**

**str2 = str2.substring(str2.indexOf(",") + 1);**

**}**

**return str1 + " " + str2 + " " + mystery(str1, str2);**

**}**

**}**

What is the output of the following statement?

**System.out.println(mystery("GO,BEARCATS", "SNOW,IS,FALLING"));**

**OUTPUT**

**BEARCATS IS,FALLING BEARCATS FALLING done**

1. (12 pts) Assume we have defined the classes and interfaces shown in the UML diagram below. Note that **Student** and **Employee** are interfaces, **AbstractStudent** is an abstract class, and all other classes are concrete.



Tell which of the following statements are legal and which are illegal. Circle the correct answer for each one.

**Student stu1 = new AbstractStudent();** Legal Illegal

**Student stu2 = new Student();** Legal Illegal

**GraduateAssistant GA1 = new Employee();** Legal Illegal

**Employee emp2 = new Graduate();** Legal Illegal

**AbstractStudent stu3 = new Undergraduate();** Legal Illegal

**GraduateAssistant GA2 = new Graduate();** Legal Illegal

1. (4 pts) Using the same UML diagram as for the previous problem, consider the following code segment.

**Student stu = new Undergraduate();**

**Undergraduate undergrad = new Undergraduate();**

**undergrad = stu;**

Which of the following statements is true about the code segment above? Circle the correct answer – be sure to circle only one choice; if you circle more than one choice, you answer will be considered incorrect.

The code segment will not compile.

The code segment will compile, but there will be a runtime error.

The code segment will compile and run.

1. (6 pts) Find the output of the following code:

**int num1 = 100;**

**int num2 = 70;**

Output

**77 45 62**

**54 20 31**

**34 0 15**

**int num3 = 125;**

**int num4 = 3;**

**while(num1 + num2 > 50)**

**{**

**num3 = num3 / 2;**

**if(num1 + num2 > 100)**

**{**

**num1 -= 3;**

**num2 -= 5;**

**}**

**for(int i = 1; i < num4; i++)**

**{**

**num1 -= 10;**

**num2 -= 10;**

**}**

**System.out.println(num1 + " " + num2 + " " + num3);**

**}**