|  |  |  |
| --- | --- | --- |
| **CIS 296** | **Final (50)** | **Winter 2010** |

I avow that I neither received nor given help on this exam \_\_\_\_\_Karen Kenward\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (type your name)

### Instructions

Please use a different COLOR for your answers, so that I can see them more clearly. You can type your answers right in this document or use another document if you wish. A question’s value in points is listed in parenthesis before the question. Questions left blank will receive 0 score. Putting something down may increase your score.

*You are to work alone on this exam.*

This assignment is individual and you are not to collaborate with anybody else. You may use your book, your notes and your compiler. You may *not* use the web (bonus section is the exception to this rule). No cheating will be tolerated.

### Exam Questions

1. (3) What is the output of the following code? Before testing this on a compiler, what do you think the output will be? Did the reality match your expectations? Explain.  
    **import** java.math.BigInteger;

**public** **class** Test

{

**public** **static** **void** main(String[] args)

{

BigInteger x = **new** BigInteger("3");

BigInteger y = **new** BigInteger("7");

x.add(y);

System.*out*.println(x);

}

}

It would seem like the output should be 10

Testing on a compiler shows the output as 3

x.add() method returns a biginteger, but it does not affect the value of x. BigInteger z = x.add(y); would output 10 , but, in this example, the return value of add() is lost.

1. (3) Agile Development concept tells you to “Fail Faster”. What does this mean and why would you want to fail and do it faster to boot?

Fail faster has to do with how much time is invested on a project that may fail. You want to work on the hardest part of the problem first. If you can’t get that to work, then there is no point spending time or resources to make the easy things work. The faster you get to the point of failing, the sooner you can cut your losses.

1. (2) What happens if you add the same component (like a JButton) twice to a JPanel?

To add more than one JButton to a JPanel, you would need two instances of JButton. Adding a single instance of JButton more than once to a JPanel will cause the button to only appear in the last position in which it was added.

1. (3) Can a source have multiple listeners? Can a listener listen on multiple sources?  
   (This is from Chapter 15 - Event-Driven Programming and also Chapter 32 - Java Bean Events)

A source may have multiple action listeners added to it. A listener may be registered to multiple sources, as well.

1. (3) How can an applet starts executing when it doesn’t have a main() method?

Applets are designed to be invoked from a Web browser, so they do not need a main(). Instead, they need a constructor with no arguments and the methods init(), start(), stop(), and destroy(). The browser creates an instance of the applet, then calls init() and start().

1. (3) If your code can throw an exception, what are the two way to deal with it?  
   (Recall that Java forces you to deal with Checked Exceptions. How? Aside from the book, Eclipse may help you if you are stuck)

The program can catch the exception and implement code to handle the exception and continue with the program. The program could also re-throw the exception and have the calling application handle the exception. For example, a file operation is an exception checked for by the compiler. The program must declare the method with “throws FileNotFoundException”.

1. (10) Write a Comparator method for this program to work as expected.  
   See comments specifying Task 1 and Task2 you are to complete.  
   See Listing 22.4, p 713 and 22.5, p 715 for reference.

**public** **class** Apple {

String name;

Integer tasteScore;

**public** Apple(String n, Integer score) {

name = n; //apple variety

tasteScore = score; //taster's score

}

**public** String toString() {**return** name;}

**public** String getName() {**return** name;}

**public** **void** setName(String name) {**this**.name = name;}

**public** Integer getTasteScore() {**return** tasteScore;}

**public** **void** setTasteScore(Integer tasteScore)

{**this**.tasteScore = tasteScore;}

}

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

//Task 1: create a TreeSet of type Apple and define a new Comparator called AppleOrder

Set<Apple> set = **new** TreeSet<Apple>(**new** AppleOrderComparator());

set.add(**new** Apple("McIntosh", 4));

set.add(**new** Apple("Braeburn", 3));

set.add(**new** Apple("Jonagold", 2));

//we want to print out apple names in order of the taste score,

// lowest to highest i.e. [Jonagold, Braeburn, McIntosh]

System.*out*.println(set);

}

}

//Task 2: Write the code for the AppleOrder Comparator class below. Compare apples by the tasteScore.

import java.util.Comparator;

**public** **class** AppleOrderComparator **implements** Comparator<Apple>, java.io.Serializable {

**private** **static** **final** **long** *serialVersionUID* = 1L;

**public** **int** compare(Apple a1, Apple a2) {

**int** score1 = a1.getTasteScore();

**int** score2 = a2.getTasteScore();

**if** (score1 < score2)

**return** -1;

**else** **if** (score1 == score2)

**return** 0;

**else**

**return** 1;

}

}

1. (3) What happens to a synchronized container if a thread is accessing it via an iterator while the container is being modified by another thread? (See page 961 for a fail-fast iterator concept)

If a synchronized container is accessed by an iterator while the container is being modified by another thread, the iterator will immediately fail and throw a ConcurrentModificationException. The iterator needs to acquire a lock on the synchronized container when it is going to traverse the container in order to avoid this error.

1. (10) Design a small programming exercise worth about 10 points in difficulty. The exercise should be computationally interesting. In other words, it should have some purpose.

The answer to this question is a textual write-up explaining the problem statement. Length of the write up can be as short as a sentence or two, or as long as a paragraph or two. Include specific examples for clarity. Do not include any code for this question as it will not be considered for grading. Thinking about how the problem may be coded up, however, may help you when designing the statement.

### More Info

Your exercise should be a problem statement that takes in some input, does some processing and produces some output. Incorporate two or more concepts in your exercise. By concepts I am talking about things like arrays, strings, numbers, sorting, searching, finding max and finding min, finding specific values, testing values for a certain condition, counting, adding and removing things from containers, and so on. Specify the concepts you are using if they are not immediately clear.

## Answer Space

Write your problem statement here: Use an extra page if needed.

Write a Java application demonstrating Inheritance and Polymorphism. Create a Robot Class with properties for Name, Speed and Health and 2 virtual methods. One virtual method for speed over rocky terrain will decrement the robot’s speed by 70%. The other, for damage from bullets, will decrement health by 45%. Create 2 classes that will inherit from Robot Class, say Terminator class and Johnny5 class. Each will have to override the methods of Robot. Terminator’s speed over rocky terrain decrements by 30% and damage from bullets decreases health by 30%. Johnny5’s speed over rocky terrain decrements by 80% and damage from bullets decreases health by 60%.

The application should read name, robot type, speed, and health from an input file, create instances of the appropriate robot and store all robots in one array. Iterate through the robot array and call the appropriate methods and display results for each.

1. (10) Write a program that computes the number of divisors for any given positive integer N.  
   Use Java’s Map container in your program. Not using Java’s Map may decrease your score.

For example, divisors of 12 are 1, 2, 3, 4, 6, and 12. Thus, 12 has 6 divisors.

### Algorithm to use for Map Container:

* Write a method that generates all prime factors of N.  
  For 12, the factors are 2, 2, and 3, since 2 \* 2 \* 3 = 12.
* Count the number of times each factor shows up in the listing.  
  This is the same as powers for each individual factor:



* Increment the powers of the unique factors by one and multiply them together.  
  (2 + 1) \* (1 + 1) = 6 divisors.

### Formula:



### Sample Run:

* 2 is generated as a factor of 12. It is added to the map. Since this is the first key == 2 in the map, this (key, value) pair is added: (2, 1). Here, key is a unique factor, and value is the number of times that factor shows up in the factor listing of a number.
* Second factor of 2 is generated. Since map already contains a key of 2, we first find out the value of that key, which is 1. Then we increment it by 1, and insert new (key, value) pair into the map. (2, 2)
* 3 is generated, and we add (3, 1) to the map.
* At the end, we iterate through the map by going through each key. As we go we produce a product by multiplying together keys’ values incremented by one.

**Question 10 Answer Space** Paste your Java code here:

**import** java.util.\*;

**public** **class** Test {

**public** **static** Map<Integer, Integer> *divisorsMap* = **new** HashMap<Integer, Integer>();

**public** **static** **void** main(String[] args) {

**int** n = 12;

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

primes = *findPrimes*(n);

**for** (Integer i : primes) {

**int** quotient = n;

**while** (quotient % i == 0) {

*addToMap*(i);

quotient /= i;

}

}

Set<Map.Entry<Integer, Integer>> entrySet = *divisorsMap*.entrySet();

Integer result = 1;

**for** (Map.Entry<Integer, Integer> entry : entrySet) {

result \*= entry.getValue() + 1;

}

System.*out*.println(*divisorsMap*);

System.*out*.println(result);

}

**public** **static** **void** addToMap(Integer divisor) {

**if** (*divisorsMap*.get(divisor) == **null**)

*divisorsMap*.put(divisor, 1);

**else** {

**int** value = *divisorsMap*.get(divisor).intValue();

value++;

*divisorsMap*.put(divisor, value);

}

}

**public** **static** ArrayList<Integer> findPrimes(**int** number) {

// for each number, test if it is prime

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

**int** end = number / 2;

**for** (Integer i = 2; i <= end; i++) {

**if** (*isPrime*(i)) {

primeList.add(i);

}

}

**return** primeList;

}

**public** **static** **boolean** isPrime(Integer value) {

// try dividing by each number between 2 and half the value

**for** (**int** i = 2; i <= value/2; i++) {

**if** (value % i == 0) { // if it divides evenly, value is not prime

**return** **false**;

}

}

**return** **true**; // value is prime

}

}

## Bonus

1. (1) Who is the person shown at the start of every chapter in the book? What is he wearing?

He looks like he is wearing a loin cloth or something a sumo wrestler might wear. Maybe he is a Sumo wrestler in training. Or perhaps he is a servant from the Towers of Hanoi, which is also a popular puzzle to solve with recursion.

1. (1) Who is James Gosling? Why am I asking about him?

James Gosling led a team at Sun Microsystems. They developed Java to be an internet programming language.