### Name:\_\_\_\_\_\_\_\_\_\_\_\_\_

**Application Engineering and Development INFO 5100**

# Exam I

## Friday Feb 15, 2013

1. Consider the model below and the vital signs table (attached) on one of the following slides Define a method on the patient class called isPatientBloodPressureAbnoralNormal(). The method returns true or false based on the rules in the vital signs table. Your job is to define two java classes with all necessary attributes to fulfill the requirements of the rules on the vital signs table. To test your java implementation show how to instantiate instances to test each of the six two of the age groups presented in the table. In other words create objects for at least 2 patients in different age groups; assign them vital sign values in all the ranges and use the System.out.printlin to confirm that your program is working correctly.
2. [Vital Sign History] During a visit to the doctor’s office (called an encounter), a patient’s vital signs are captured. This happens every the patient goes to see a doctor. To understand the patient health a history of all the vital signs over time are saved for review and assessment later. To distinguish the vital signs we must keep track of the latest vital signs by using a reference to the measurements. When the vital signs to be captured we create a new vital sign object and make it the current one. But, before we do that we must save (move) the current vital (previous) to the vital sign history list. The model below defines such an approach. Define the java class necessary for implementing the model below. Give a detailed definition of the vital sign object according to the table given (or used the one defined in the previous question). Also, provide implementation for creating and populating vital sign objects. When vital sign object is created a link to it in the vital sign history is maintained at all time. It is your responsibility to decide where these methods will go (e.g., createNewVitalSign(). )

1. Consider the model below.
   1. Implement the following method: findAllPatientsWithAbnormalHighBloodPressure(). This method searches through all patients with current vital signs that appear abnormal. It returns an arraylist of all such patients. Provide a full java implementation of the model below. Define test cases to prove that your java is working correctly.
   2. Implement a similar method to return abnormal blood pressure of people of certain age group findAllPatientsWithAbnoralHighBloodPressureByAgeGroup(age) where age is the input. You can assume age is an int if you want.
2. Product management personnel maintain the company’s product catalog on an ongoing basis. For example, sometimes they want to update the product definition in terms of prices, description, etc. To make changes, management requires that old versions of the product definition must be preserved. So if one is to make a change to a product, a new copy of the current product is made and the current product is saved as old. The following class definition has some methods that allow you to create versions of the product which are then maintained part of the product definition. Extend the class definition below by defining a subclass called Product, adding attributes to keep track of the person making the change as well as adding the update date (use the System.currentTimeMillis() method which will return the time as long. Store it and displayed as long for the purposes of this exercise). Make sure to fully implement all the abstract methods in the Product class using proper java syntax.

public abstract class AbstractProduct {

String name;

int price;

private AbstractProduct previousversion; //this ref variable keeps track of the earlier version of the product

public abstract AbstractProduct(){

}

Public abstract AbstractProduct updateProduct();

public void setName(String n){

name = n;

}

public void setPrice(int p){

price = p;

}

public AbstractProduct getPreviousVerson(){

return previousversion;

}

public abstract ArrayList<AbstractProduct> getProductHistory(); //returns all previous copies (versions) of the product.

}

1. What does this class do? How to use it and in what situations? Please explain your answers.

Class Business

{

static Business business = null;

static Business GetInstance()

{

if(business == null)

business= new Business()

return business;

}

}