National University of Computer and Emerging Sciences
Karachi Campus

Object-oriented Programming (CS-1004)

Quiz-3 and 4 Exam

Date: 6/5/2025

Course Instructor(s)

Minhal,

Total Time: 1hr

Total Marks: 100

Total Questions: 01

Roll No _____ Section _____ Student Signature _____

Do not write below this

The HeatStrokeCampSystem (HSCS) is a Java-based object-oriented application designed to manage medical relief operations during extreme heat wave conditions in Karachi . The system registers patients, assigns medical staff (doctors and volunteers), records vital statistics, identifies high-risk hotspots, and performs file-based data handling while demonstrating core OOP principles.

The abstract class Person declares encapsulated attributes name:String and id:int, with corresponding getter and setter methods getName():String, setName(String):void, getId():int, and setId(int):void, along with an abstract method displayInfo():void.

Two concrete classes, Patient and MedicalStaff, extend Person. The class Patient includes attributes campLocation:String, a composed object vitals:Vitals, and an aggregated reference assignedDoctor:Doctor. It implements the interface VitalsRecordable and provides methods like updateVitals(double):void, getVitals():Vitals, and getCampLocation():String.

The abstract class MedicalStaff adds the attribute specialization:String and defines the abstract method provideCare(Patient):void. It will be extended by two concrete subclasses Doctor and Volunteer, both of which override the method provideCare(Patient):void, demonstrating dynamic polymorphism (method overriding).

The class Vitals encapsulates vital statistics, especially temperature:double, and provides overloaded methods setTemperature(double):void and setTemperature(double, boolean):void to handle values in Celsius or Fahrenheit, along with a getter getTemperature():double, thus demonstrating static polymorphism (method overloading).

The interface VitalsRecordable declares a method updateVitals(double):void, which is implemented in the Patient class.

The CampManager class manages system coordination and includes methods such as assignDoctorsToCamp(String,Doctor[]):void,getCriticalPatients(Patient[],double):Patient[],andaddVitalsAnd Filter(ArrayList<Patient>,double):ArrayList<Patient>, demonstrating functional dependency by accepting and returning arrays and ArrayList of objects.

The FileOperations interface declares methods for object-oriented file handling: saveToFile(String,ArrayList<Patient>):void,loadFromFile(String):ArrayList<Patient>,andfilterHotspotPatients (ArrayList<Patient>,double,int):ArrayList<Patient>.

These are implemented by the PatientFileHandler class using Java serialization. The method filterHotspotPatients(...) by manually maintaining parallel ArrayList<String> (for camp locations) and ArrayList<Integer> (for case counts). It first loops through the patients to count the number of cases per

location, then filters patients with temperature > tempThreshold and whose campLocation has case count greater than caseThreshold.

Custom exception classes include InvalidTemperatureException, thrown by setTemperature(...) in Vitals if temperature is out of acceptable range,

NullDoctorAssignmentException, thrown in CampManager when attempting to assign a null doctor, and DataFileEmptyException, thrown in file handler classes if the loaded data is empty.

The main() method of the **Heat Stroke Camp System (HSCS)** simulates a heatwave emergency response scenario where the system must register multiple Patient objects, update their Vitals, assign Doctor objects, and perform filtering to identify high-risk heatstroke hotspots based on temperature and number of patients per location. The program demonstrates key OOP concepts: dynamic polymorphism via Doctor and Volunteer implementing provideCare(), method overloading in Vitals, custom exceptions like InvalidTemperatureException and NullDoctorAssignmentException, and functional dependency by passing arrays and ArrayList of objects in method parameters and return types. It also shows object-oriented file handling where a list of patients is saved to and loaded from a .dat file, and filtered using custom logic to detect hotspots.