BITS, Pilani – KK Birla Goa Campus
Department of CS & IS
CS/IS F213 Object Oriented Programming
IS C313 Object Oriented Programming and Design

17.09.2014 Test – I

Max. Time: 60 min Max. Marks: 45

Instructions:

1. Answer all parts of a question at one place.
2. Write clear answers for complete credit.
3. Make an index on the back side of the front page of answer booklet.

Question-1

ICC One-Day Cricket World Cup 2015 is to be played in Australia and New Zealand. You have been chosen as software developer for creating software to manage the cricketing aspects of the tournament; other parts of the tournament such as ticket sales, ground/venue management, live telecast, hospitality, tourism and promotion are not part of your development mandate. Using your understanding of object oriented concepts, answer the following questions.

A. Identify hierarchy, key abstractions for the software system. **(5M)**
B. Draw class and object hierarchy. **(5M)**
C. It is said that primitives at each level of hierarchy are relative to observer. Identify relative primitives as seen by Umpire, Team Captain. Point out the differences. **(5M)**
D. Draw CRC card for a cricketer. **(4M)**

Question-2

Write a Java program to iterate through a set of sentences and identify the sentences containing the words smalltalk or SmallTalk. Print the sentences that contain the given words. The program must have the following elements.

A. Write a regular expression to identify the given words. **(2M)**
B. Write the complete Java program. **(6M)**
C. Write proper Java documentation for your code. **(3M)**

Question-3

Read the sample Java code given below.

```
package org.jnetpcap.examples;
import java.io.File;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.List;

import org.jnetpcap.Pcap;
import org.jnetpcap.PcapDumper;
import org.jnetpcap.PcapHandler;
import org.jnetpcap.PcapIf;


public class PcapCapture implements Serializable, List<PcapPacket> {
  public static void main(String[] args) {
    List<PcapIf> alldevs = new ArrayList<PcapIf>();
    StringBuilder errbuf = new StringBuilder();

    int r = Pcap.findAllDevs(alldevs, errbuf);
    if (r == Pcap.NOT_OK || alldevs.isEmpty()) {
      System.err.printf("Can't read list of devices, error is %s\n", errbuf.toString());
```

Scanned by CamScanner

```
    return;
  }
  PcapIf device = alldevs.get(1);

  Pcap pcap = Pcap.openLive(device.getName(), 64*1024,
                            Pcap.MODE_PROMISCUOUS, 10 * 1000, errbuf);
  if (pcap == null) {
    System.err.printf("Error while opening device for capture: %s\n",
      errbuf.toString());
    return;
  }

  String ofile = "tmp-capture-file.cap";
  PcapDumper dumper = pcap.dumpOpen(ofile);

  ... //irrelevant lines

  File file = new File(ofile);
  System.out.printf("%s file has %d bytes in it!\n", ofile, file.length());

  dumper.close();
  pcap.close();

  if (file.exists()) {
    file.delete();
  }
 }
}
```

A. Draw sequence diagram for the code given. **(7M)**

B. Draw the class and object diagrams for the code given. Clearly show the roles, visibility, variables and methods outlined in the code given above. **(8M)**

C. Draw package diagram for the given code. **(5M)**

Scanned by CamScanner

BITS, Pilani – KK Birla Goa Campus
Department of CS & IS
CS/IS F213 Object Oriented Programming
IS C313 Object Oriented Programming and Design

22.10.2014 Test – II

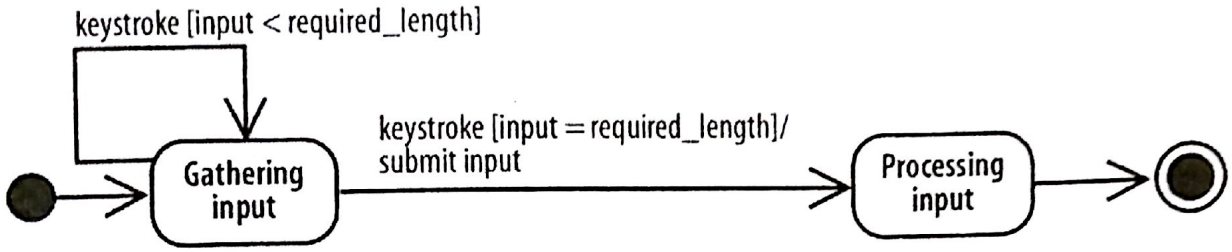
Max. Time: 60 min Max. Marks: 45

Instructions:

1. Answer all parts of a question at one place.
2. Write clear answers for complete credit.
3. Make an index on the back side of the front page of answer booklet.

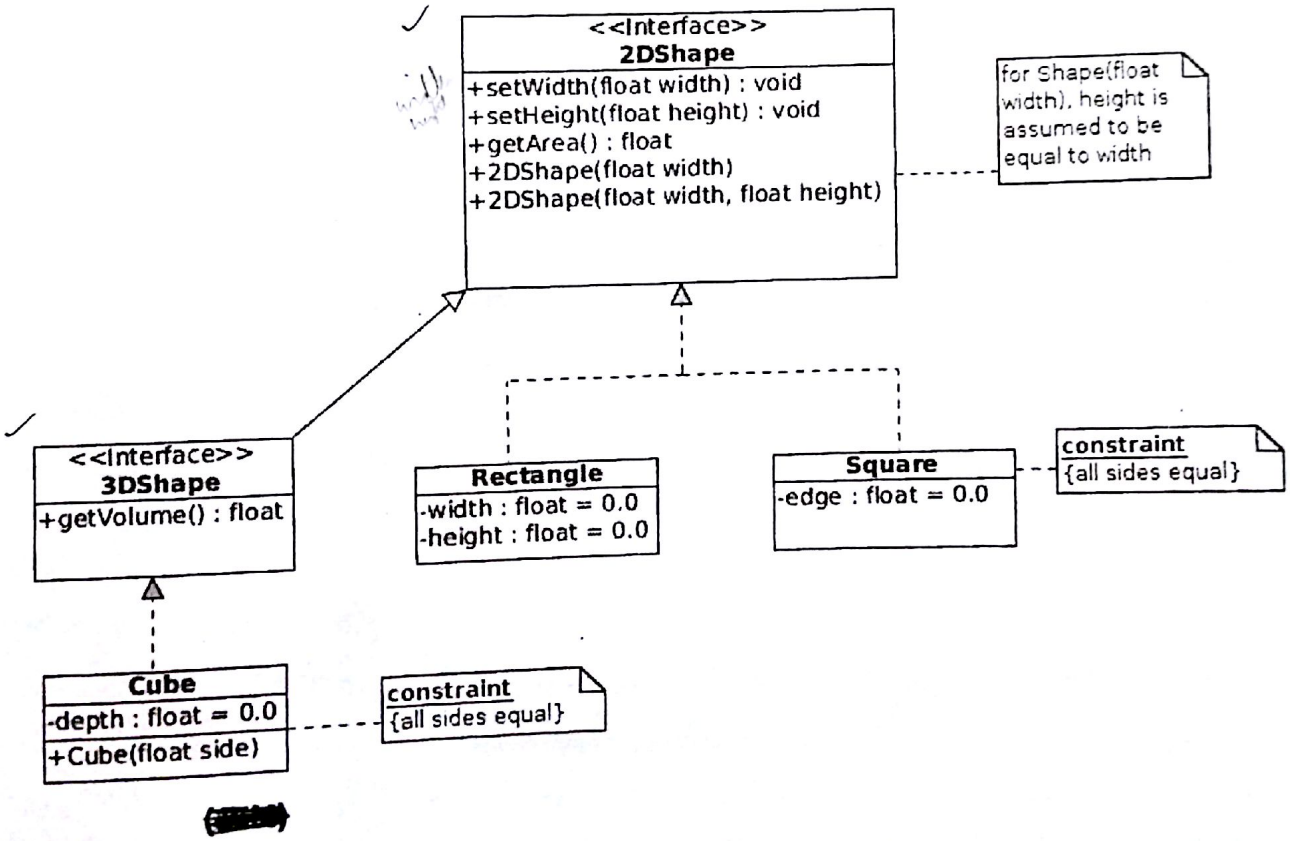
Question-1

Write a Java programe to implement the following input processing state diagram for a keyboard driver. **(15M)**



Question-2

A. Write a Java programme to implement the UML class diagram given below. (15M)



B. Refactor the class diagram to improve the naming conventions and inheritance semantics. Justify your changes. **(7M)**

Scanned by CamScanner

Question-3

A Java program to read contents of a file and sum up the values is given. Modify the code to anticipate `FileNotFoundException`, `IOException` and `BadDataException` from readFile() method of **DataSetReader** class. The next() method of **Scanner** class can also throw `NoSuchElementException` and `IllegalStateException` exceptions.

All the stated exceptions have a constructor that takes ***String*** *message* as a single arguement. In the *try...catch...finally* block, print the exception message using getMessage() method of these exception classes. **(13M)**

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Scanner;

public class DataAnalyzer
{
  public static void main(String[] args)
  {
    Scanner in = new Scanner(System.in);
    DataSetReader reader = new DataSetReader();

    boolean done = false;
    while (!done)
    {
        System.out.println("Please enter the file name: ");
        String filename = in.next();

        double[] data = reader.readFile(filename);
        double sum = 0;
        for (double d : data) sum = sum + d;

        System.out.println("The sum is " + sum);
        done = true;
    }
  }
}
```

Scanned by CamScanner

BITS, Pilani – KK Birla Goa Campus
Department of CS & IS
CS/IS F213 Object Oriented Programming
IS C313 Object Oriented Programming and Design
Comprehensive Exam

NC

11.12.2014 Max. Time: 3 Hours Max. Marks: 105

Student ID: 2013A7PS045G Student Name: Varsheeth Tallurri

Marks

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Total
2	0	12	10	10	11	20	15		80

Recheck Request

Please recheck Q 3 & Q 6. It is all ticked but I don't understand where I was wrong.

NC

Q.3 class diagram is incomplete

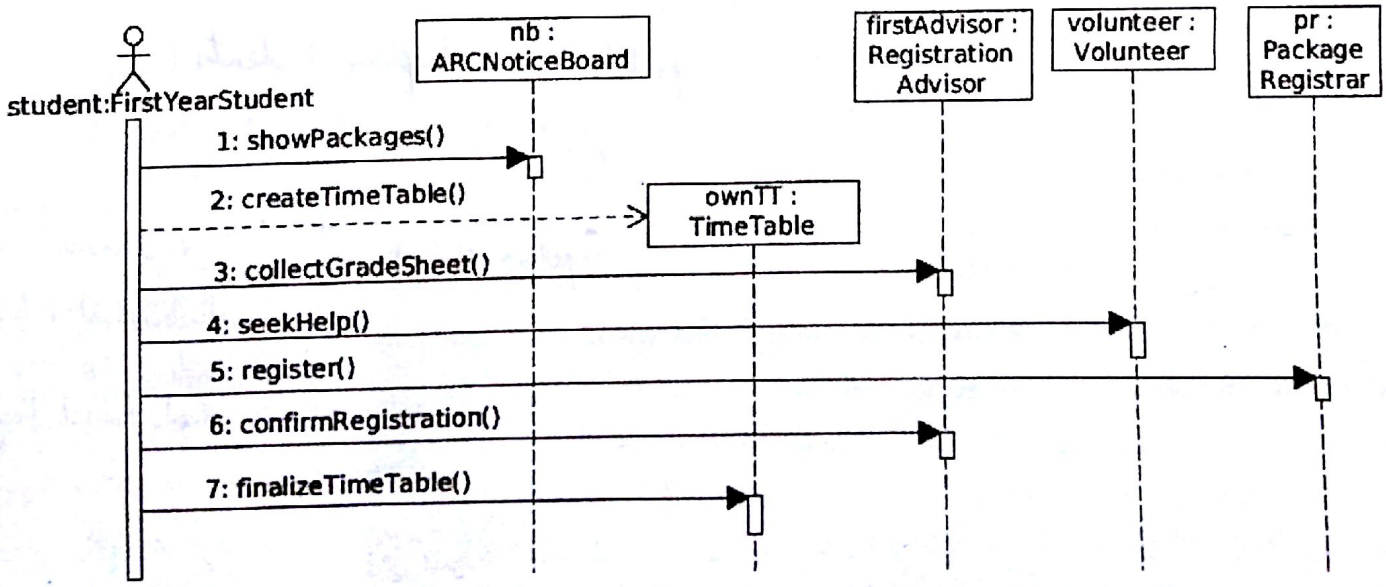
Q.6 you haven't considered case for -ve values

Question-1 (Answer on the back side of this sheet)
Write the *readStudents*, *displayStudents*, *lookUpStudents* methods for **ParticipantList** class. **(15M)**

```
import java.util.*;
public class ParticipantList {
      private HashMap<String,String> studentsList =
                  new HashMap<String,String>();
      public static void main(String[] args) {
            ParticipantList list = new ParticipantList();
            Scanner sc = new Scanner(System.in);
            list.readStudents(sc);
            list.displayStudents();
            list.lookUpStudents(sc);
            sc.close();
      }
```

Question-2 (Answer on the back side of this sheet)
A tentative sequence diagram for semester registration of first year students is shown. Modify the diagram to incorporate the following changes. **(10M)**

1. A clash in time table will be indicated by **Volunteer** with a *clash* return message as a response to message 4. If the time table is clash-free, **Volunteer** returns Ok.
2. **PackageRegistrar** may either return a *registered* or *full* message. If student gets a *full* message, student needs to sit down with volunteer to select another package. Appropriate methods that will be useful are: *ownTT.alterTT*, *volunteer.seekHelp*, and *pr.register*. These steps are to be repeated until the registration is successful.
3. When a package is full, sends an asynchronous **Whatsapp** message to all friends informing them of the package status. The update is done by invoking *sendMessage(friends, "Package #no is not available anymore")* on **wapp:WAClient** object.



Scanned by CamScanner

Question-3 (Draw the CRC card and class diagram in the space given below; draw the object diagram on the back side)
Diversity is a dance group from UK; they seem popular now. Lets say you are one of the organizers of Waves-2015 and chose to invite Diversity for a music night. Diversity group members have personal information in the form of name, date of birth and twitter handle. For the music night, diversity shall perform four dances. Each dance shall have a lead dance pair followed by backup dancers. The lead dance pair will be different for each dance performance. Diversity group cosists of

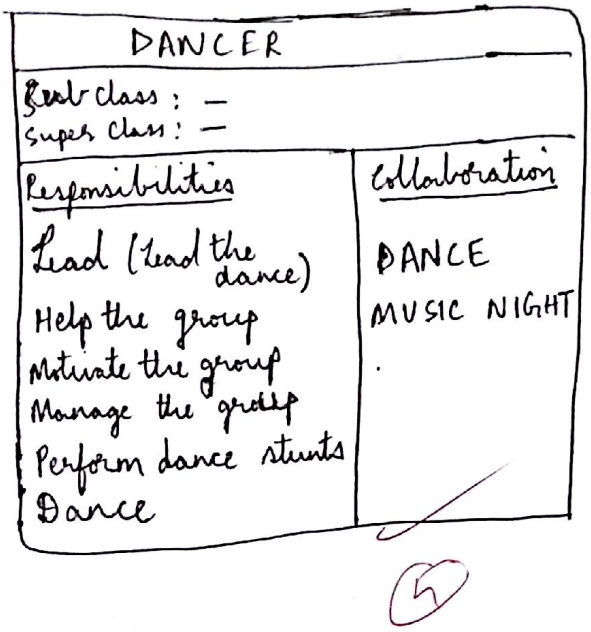
Ashley Banjo Born: 4 October 1988 @AshleyBanjo	Jordan Banjo Born: 31 December 1992 @Jordan_Banjo	Sam Craske Born: 7 September 1990 @Sam_Craske	Mitchell Craske Born: 27 April 1996 @Mitch_Craske
Ike Ezekwugo Born: 7 October 1986 @IkeTheKidd	Perri Kiely Born: 15 December 1995 @realperrikiely	Warren Russell Born: 3 May 1990 @Warren_DV	Terry Smith Born: 12 February 1985 @Tel_DV

The first dance of night will be performed by Banjo brothers as lead pair and rest of the crew as backup dancers. The second dance shall be performed by Ike Ezekwugo and Sam Craske in the lead with the rest of dancers taking on the role of backup dancers.

1. Draw the CRC card of a dancer.
2. Create class diagrams that will capture the personal details of each dancer, group structure of each dance and that of the music night.
3. Draw object diagram for the first and second dances.

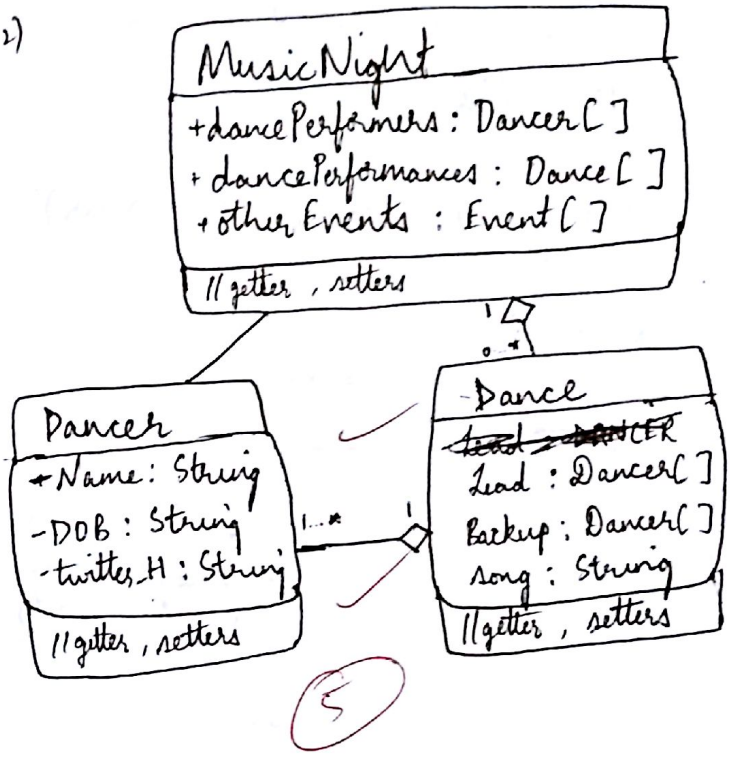
(20M)

1)



5

2)



5

Question-4 (Answer on the back side of this sheet)
Create a class called **Dog** that has three fields: String name, Person owner, and int age.
Create a constructor for **Dog** that takes a String name and a Person owner for the Dog and uses them for initialization. Have the age for a Dog start at 0;
Implement getter and setter methods for both name and owner. Make the mutator for name such that whenever a name is applied to a Dog, the actual name of the Dog is "<Given Name> the Dog". (e.g. given "Tony", the Dog's name becomes "Tony the Dog").
Implement only an getter for age. Implement a method called haveBirthday that does not return anything and simply increases a Dog's age by one.
Finally, write a method called toString that returns a string of the form: "< Dog name> is <age> and belongs to <Owner name>" e.g. "Tony the Dog is 6 and belongs to Rahul"
The definition for Person is found below.

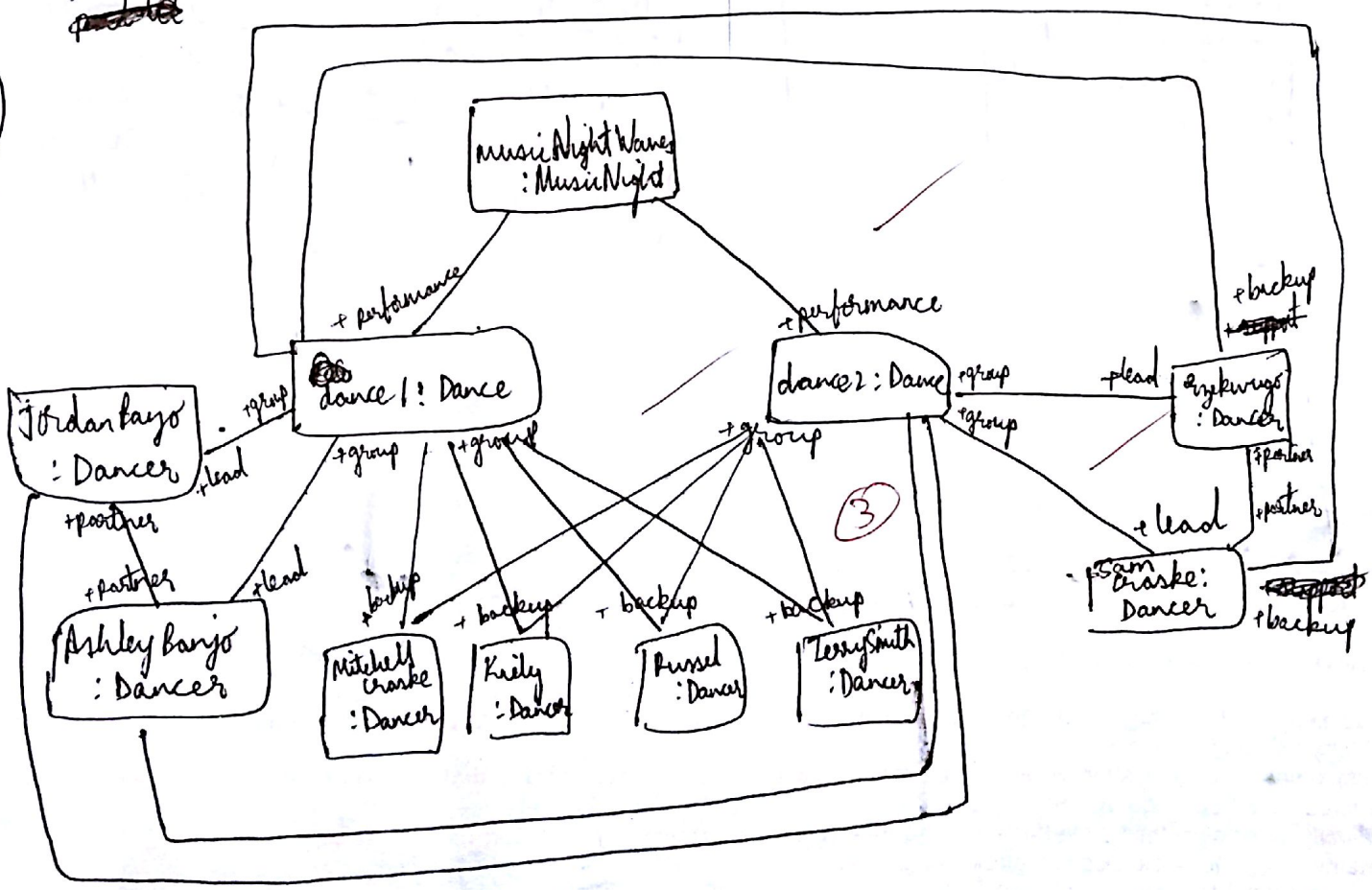
(10M)

```
public class Person {
    private final String name;
    public Person(String newName) {name = newName;}
    public String getName() {return name;}}
```

Scanned by CamScanner

4)
```
public class Dog {
    private String name;
    private Person owner;
    int age;
    public Dog( String name, Person owner){
        this.name = name;
        this.owner = owner
        this.age = 0;
    }
    public void setName(String name){
        this.name = name + " the Dog";
    }
    public String getName(){
        return this.name;
    }
    public void setOwner(Person Owner){
        this.owner = owner;
    }
    public Person getOwner(){
        return this.owner;
    }
    public int getAge(){
        return this.age;
    }
    public void haveBirthday(){
        this.age += 1;
    }
    public String toString(){
        String str = name + "is" + age "and belongs to"
            + owner.getName();
```

3)



Scanned by CamScanner

Question-5 (Write your answer in the space provided below)

Implement a **Bicycle** class which has the following three methods:

```
public void increaseSpeed();
public void decreaseSpeed();
public boolean isMoving();
```

Within the **Bicycle** class, you must keep track of the Bicycle's state: moving or not moving. You must also keep track of the Bicycle's current speed. Whenever the Bicycle has a positive current speed, the state should be moving. Whenever the Bicycle has a current speed of 0 the state must be not moving.

You must use a boolean variable to maintain the Bicycle's state. The methods *increaseSpeed* and *decreaseSpeed* always increment and decrement [respectively] the current speed by 1. If *increaseSpeed* is called on a Bicycle which is not currently moving, the Bicycle should be set to moving, and the current speed should be increased by 1. If *decreaseSpeed* is called on a Bicycle which is moving, the current speed should be decremented by one. If the current speed is ever decreased to 0, the Bicycle's state should change from moving to not moving.

The method *isMoving* should return the status of the Bicycle.

Your Bicycle class should also provide two constructors. One constructor takes no arguments and the other takes an integer representing the initial speed of the Bicycle. The default constructor should create a Bicycle which is not currently moving and has a current speed of 0. The second constructor should set the current speed to the passed initial speed ONLY if the speed is positive. It should also set the Bicycle's state to moving. If the initial speed given is negative or 0, the current speed should be set to 0 and the Bicycle's state should be not moving. **(10M)**

```
public class Bicycle {
    public boolean state; // state = true if moving, false if not.
    public int speed;

    Bicycle (int speed){
        if(speed > 0)
        { this.speed = speed;
          this.state = true;
        } else if (speed <= 0) {
            this.speed = 0;
            this.state = false;
        }
    }
    Bicycle ( ){
        this (0);
    }
    public void increaseSpeed ( ){
        if(this.speed == 0)
        {
            speed += 1;
            state = true;
        } else if (this.speed > 0){
            speed += 1;
        }
    }
    public void decreaseSpeed ( ){
        if(this.speed <= 1)
        {
            speed -= 1;
            state = false;
        } else if (this.speed > 1) {
            speed -= 1;
        }
    }
    public boolean isMoving ( ) {
        return this.state;
    }
}
```

10

Question-6 (Answer on the back side of this sheet)

Write a jUnit test suite which would test increaseSpeed() and isMoving() methods of Bicycle class specified in ***Question-5***. The behavior of both the methods has been described in ***Question-5***. **(15M)**

Scanned by CamScanner

6) import junit.framework.*;

```
public class TestBicycle extends TestCase{
public TestBicycle (String name){
        super(name);
    }
public void testIncreaseSpeed(){
    Bicycle b1= new Bicycle();
    assertEquals(0, b1.speed);
    Bicycle b2 = new Bicycle(5);
    assertEquals(5, b2.speed);
    b2.increaseSpeed();
    assertEquals(6, b2.speed);
    b1.increaseSpeed();
    assertEquals(1, b1.speed);
    assertTrue(b1.state);
    Bicycle b3 = new Bicycle();
    assertFalse(b3.state);
}
public void testIsMoving(){
    Bicycle b1 = new Bicycle();
    assertFalse(b1.isMoving());
    Bicycle b2 = new Bicycle(6);
    assertTrue(b2.isMoving());
    b1.increaseSpeed();                 //bicycle starts moving
    assertTrue(b1.isMoving());
    b1.decreaseSpeed();                 //bicycle stops
    assertFalse(b1.isMoving());
}
public static Test suite(){
    TestSuite ts= new TestSuite();
    ts.addTest(new TestBicycle("testIncreaseSpeed"));
    ts.addTest(new TestBicycle("testIsMoving"));
    return ts;
}}
```

11

Scanned by CamScanner

Question-7 (Answer on the back side of this sheet)

Several design properties for a problem are presented below. Use these properties in order to write all the necessary classes and/or interfaces for a solution to the problem. Focus on class structure and interaction. You may implement your solution however you wish, but you will be evaluated on the appropriateness of your solution to the requirements. Note the use of capitalization and parentheses for clarification. You may use whatever constructors or additional methods you wish.

You must define a structure that can represent Animals. Animals have two behaviors; they can speak() and they can move(). By default, when an animal moves, the text "This animal moves forward" is displayed. By default, when an animal speaks, the text "This animal speaks" is displayed. A general Animal should not be able to be instantiated.

Define also two classes, Goose and Lynx, that are Animals. Both Goose and Lynx behave such that where "animal" is displayed in speak() or move(), "goose" or "lynx" is displayed by the appropriate classes.

Finally, any instance of Goose can fly(), just as any Flying object can. An Airplane is also a Flying object. Define the Airplane class such that it is Flying and make sure that any instance of Goose is also Flying. The specific behaviors when instances of either class fly() are up to you. Instances of either Goose or Airplane should be able to be stored in a variable of type Flying. **(20M)**

```
1) public abstract class Animals{
    protected String name = "animals";
    public void speak(){
      System.out.println("This" + this.name + "speaks");
    }
    public void move(){
      System.out.println("This" + this.name + "moves");
    }
  }

2) public interface Flying{
    public void fly();
  }

3) public class Lynx extends Animals{
    protected String name = "Lynx";
  }

4) public class Airplane implements Flying{
    public void fly(){
      System.out.println("This aeroplane flies");
    }
  }

5) public class Goose extends Animals
      implements Flying{
    protected String name = "Goose";
    public void fly(){
      System.out.println("This goose flies");
    }
  }
```

20

Question-8 (Answer on the back side of this sheet)

As part of a wild life tracking software development, you need to store and retrieve locations of animals. Each spotted animal is represented as object of type **Animal** having attributes *animal ID, animal type, lattitude and longitude values.* It is known that the animal ID is unique among all the animals. *The* spotted animals are to be stored in the **LastSpotted** table of **WildLifeTracking** database. A typical row that is stored in the table is,

Z1053, Zebra, -3.06646, 37.35067

Assume the existence of **WildLifeTracking** database and that user *oop* has access to the data; the password for the user *oop* is *passwd*. Write a **AnimalDBInterface** class which would persist locations of spotted animals into the database. You can use *CREATE TABLE IF NOT EXISTS* to optionally create the **LastSpotted** table for the first time. The table is to be created in the constructor and any exceptions thrown must be caught. Also develop methods *storeLocation(Animal animal)* and *fetchLocation(String animalID)* that would interact with the database. **(15M)**

Scanned by CamScanner

2)

```
import java.sql.*;
public class AnimalDBInterface {
    public static final DRIVER = "org.h2.Driver";
    public static final URL = "WildLifeTracking";
    public static final USER = "oop";
    public static final PASSWD = "password";

    AnimalDBInterface ( ) {
        Connection conn = null;
        Statement stmt = null;
        try { Class.forName(Driver);
            Conn = DriverManager.getConnection
                                (URL, USER, PASSWD);
        stmt = Conn.createStatement ();
        String sql = "CREATE TABLE if not exists
                    LastSpotted (
                        animalID char[10] is unique,
                        animalType char[64],
                        latitude float,
                        longitude float ) ";
        stmt.executeQuery(sql); stmt.close(); conn.close();
        } catch (SQLException se){
            se.printStackTrace ();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (conn != null)
                conn.close();
            if (stmt != null)
                stmt.close();
        }
    }
    public void storeLocation (Animal animal){
        Connection conn = null;
        PreparedStatement stmt = null;
        try { Class.forName (Driver);
        conn = DriverManager.getConnection(URL, USER,
                                PASSWD);
        String sql = "INSERT INTO LastSpotted
                    values (?, ?, ?, ?) ";
        stmt = conn.PrepareStatement (sql);
        stmt.setString (1, animal.animalID);
        stmt.setString (2, animal.animalType);
        stmt.setFloat ( 3, animal.latitude);
        stmt.setFloat (4, animal.longitude);
        stmt.executeUpdate ();
        stmt.close(); conn.close();
        } catch (SQLException se){
            se.printStackTrace ();
        } catch (Exception e){
            e.printStackTrace();
        } finally {
            try { if (stmt != null)
                    stmt.close();
            } catch (Exception e){ }.
            try { if (conn != null)
                    conn.close();
            } catch (Exception e){ }
    }
    public void fetchLocation(String animalID){
        Connection conn = null;
        PreparedStatement stmt = null;
        try { Class.forName (Driver);
        Conn = DriverManager.getConnection (URL, USER,
                                PASSWD);
        String sql = "Search latitude, longitude
                    from LastSpotted where
                    animalID = " + animalID;
        stmt = conn.prepareStatement (sql);
        ResultSet rs = stmt.executeUpdate ();
        String latitude = rs.getString ("latitude");
        String longitude = rs.getString ("longitude");
        System.out.println ("latitude = " + latitude +
                    "longitude = " + longitude);
        stmt.close ();
        conn.close ();
        } // catch everything as previous
                    methods.
```

15

Scanned by CamScanner