אוניברסיטת בן-גוריון

מדור בחינות

מספר נבחן:_____

רשמו תשובותיכם בגיליון התשובות בלבד
תשובות מחוץ לגיליון לא יבדקו.
שימו לב:
על תשובות ריקות יינתן 20% מהניקוד!

בהצלחה!

תאריך הבחינה: 02.03.2015
שם המורה: פרופ' אנדרי שרף
ד"ר רן אטינגר
ד"ר ג'ון מרברג
ד"ר מני אדלר
שם הקורס: תכנות מערכות
מספר הקורס: 202-1-2031
מיועד לתלמידי: מדעי המחשב, הנדסת תוכנה
שנה: תשע"ה
סמסטר: א'
מועד: ב'
משך הבחינה: שלש שעות
חומר עזר: אסור

שאלה 1 (30 נקודות)

במועד א' התוודענו לממשק Gear מגדיר גלגל שיניים ע"פ נקודות המרכז שלו (קואורדינטות x,y), אורך הרדיוס, ושני גלגלי השיניים שמצידו הימני והשמאלי. כמו כן מגדיר הממשק את המתודה move, המסובבת את הגלגל בכיוון הנתון (עם או נגד כיוון השעון), ואת הגלגלים משני צדדיו בכיוון ההפוך.

```
interface Gear {
        float getX(); // returns the x coordinate of the center
        float getY(); // returns the y coordinate of the center
        float getRadius(); // returns the length of the radius
        Gear getLeft(); // returns the gear on the left
        Gear getRight(); // returns the gear on the right
        void move(boolean clockwise); // advance the gear and both its neighbors
        void moveLeft(Boolean clockwise); // advance the gear and its left neighbor
        void moveRight(Boolean clockwise); // advance the gear and its right neighbor
}
```

ראינו גם מימוש של הממשק. בקוד להלן סונכרנה המחלקה סנכרון מלא, כדי להבטיח את נכונות ההרצה שלה בתרחישים שונים.

```
class SimpleGear implements Gear  {
        final float _x,_y,_radius;
        Gear _left, _right;
        float _state;

        SimpleGear(float x, float y, float radius) throws Exception {
                if (!check(x,y,radius))   throw new Exception("Wrong parameter values!");
                _x = x; _y = y; _radius = radius; _left = null; _right = null;
                _state = 0;
```

```
}

public float getX() { return _x; }
public float getY() { return _y; }
public float getRadius() { return _radius; }
public Gear getLeft() { return _left; }
public Gear getRight() { return _right; }

public synchronized void setLeft(Gear left) throws Exception {
        if (_left != null) throw new Exception("Left gear is already defined");
        if (!check(left. true)) throw new Exception("Wrong gear position!");
          _left = left;
}

public synchronized void setRight(Gear right) throws Exception {
        if (_right != null) throw new Exception("Right gear is already defined");
        if (!check(right,false)) throw new Exception("Wrong gear position!");
        _right = right;
}

public synchronized void move(boolean clockwise) {
        float offset = getOffset(_radius);
        if (clockwise)
                _state = (_state + offset) % 360;
        else
                _state = (_state - offset) % 360;
        if (_left != null)
                getLeft().moveLeft(!clockwise);
        if (_right != null)
                getRight().moveRight(!clockwise);
}

public synchronized void moveLeft(boolean clockwise) {
        float offset = getOffset(_radius);
        if (clockwise)
                _state = (_state + offset) % 360;
        else
                _state = (_state - offset) % 360;
        if (_left != null)
                getLeft().moveLeft(!clockwise);
}

public synchronized void moveRight(boolean clockwise) {
        float offset = getOffset(_radius);
```

```
                if (clockwise)
                        _state = (_state + offset) % 360;
                else
                        _state = (_state - offset) % 360;
                if (_right != null)
                        getRight().moveRight(!clockwise);
        }

        protected static float getOffset(float radius) {
          // calculates the offset of a gear with a given radius
          // the implementation is not relevant for the question
        }

        protected boolean check(float x, float y, float radius) {
                return (radius>0);
                //x and y can be any values – no need to check anything
        }

        protected boolean check(Gear other, boolean whichSide) {
                if (other==null) return false;
                if (whichSide && (other,getX()>=x)) return false;
                if (!whichSide && (other,getX()<=x)) return false;
                return verifyEuclidianDistance(other);
        }

        protected boolean verifyEuclidianDistance(Gear other) {
                if (other==null) return false;
                float sideX=_x-other.getX();
                float sideY=_y-other.getY();
                float distance=_radius+other.getRadius();
                return sidex*sideX+sideY*sideY==distance*distance;
        }
}
```

א. הראו תרחיש בו מערכת המורכבת ממספר גלגלי שיניים מגיעה לחבק (deadlock). (7 נקודות)

ב. פתרו את סכנת החבק שתוארה בסעיף א', בשלש דרכים:

- צמצום ומיקוד הסנכרון
- Resource ordering
- שימוש בסמפור במקום סנכרון

בכל אחת מהדרכים יש לשמור על הנכונות.
(18 נקודות)

ג. איזו מהשיטות הייתם מאמצים במקרה זה? נמקו בקצרה (5 נקודות)

שאלה 2 (30 נקודות)

בשאלה זו נמשיך את הדיון במחלקה mvector. וקטור (vector) הינו מבנה אחסון אשר מחזיק מערך (array) באופן סדרתי בזיכרון. את וקטור מגדילים (מקטינים) באופן דינאמי בעת הכנסת (הוצאת) איבר עלפי הנדרש .

insertAt הוחלט על מדיניות הקצאת זיכרון הבאה: בעת הכנסת איבר במיקום (pos)החורג מגבולות וקטור, מקצים גודל וקטור מינימאלי שהוא חזקת 2 >pos. לדוגמא, וקטור ריק בגודל 0 אליו מכניסים איבר במיקום 5 יוקצה בגודל 8. בעת הקצאה יש להעתיק את תוכן וקטור הישן.
removeAt הוחלט על מדיניות שחרור זיכרון הבאה: בעת הוצאת איבר במיקום (pos), מקטינים את גודל וקטור בחצי אם הוצאו כבר כל האיברים הממוקמים במחצית השניה של וקטור. לדוגמא וקטור בגודל 4 שהוצאו ממנו איברים במיקומים 2 ו3 יוקטן לגודל 2.
שימו לב שבשני התרחישים האיברים נשארים באותו מיקום בוקטור לפני ואחרי הפעולה

א. ממשו את פונקציות המחלקה mvector המודגשות בקו. על בנאי מעתיק לבצע העתקה עמוקה. ניתן להוסיף פונקציות עזר אך אין להוסיף שדות או לשנות חתימות של פונקציות קיימות (15 נקודות) .

```
class node{
public:
        node();
        ~node();
        virtual node* duplicate()=0;   // creates and returns a (deep) copy of this
private:
        bool _isFree;
};

class mvector{
public:
        mvector();
        ~mvector();
        mvector(const mvector &);
        void insertAt(const node & data, unsigned pos);
        void removeAt(unsigned pos);

       node* getAt(unsigned pos){return &_array[pos];}
private:
        unsigned _len;
        node[] _array;
};
```

ב. הניחו כי מימוש הפונקציות של המחלקה my_node להלן הינו תקין, וציירו את תמונת הזכרון (stack,heap,v-table) עבור הרצת קטע ה main עד ההערה במתודה foo (10 נקודות):

```
class my_node : public node {
public:
```

```
        my_node();
        my_node(const my_node & other);
        my_node(char * s);
        ~my_node();
        node* duplicate();
private:
        long _data;
};

void foo(mvector vec){
        mvector my_vec(vec);
        //*****here*****
}

void main(){
        mvector vec;
        node *n = new my_node("test");
        vec.insertAt(*n, 3);
        foo(vec);
}
```

ג. בסוף הפונקציה main הוסיפו שורה:

```
my_node *m = (my_node*)(vec.getAt(0);
```

בחר תשובה המתאימה ביותר (5 נקודות):

1. אסור לבצע downcasting בC++ ולכן זאת שגיאה.
2. אף פעם לא ניתן לבדוק תאימות casting בזמן קומפילציה
3. ניתן לבצע downcasting באופן בטוח רק אם אין מתודות virtual
4. עבור מתודות virtual מתבצע late binding בזמן ריצה
5. אף תשובה לא נכונה

שאלה 3 (30 נקודות)

בנספח למבחן מופיעה תבנית שרת ההדפסה, כפי שנלמד בהרצאות.

בסעיפים הבאים, ניתן לענות על סעיף גם אם לא עניתם על הסעיפים האחרים.

א. עדכנו את השרת כך שירוצו לכל היותר 10 ת'רדים (5 נקודות)

ב. השלימו את המחלקה MessageLengthBasedTokenizer הממשת את הממשק Tokenizer, כך שהודעה מוגדרת לא ע"י תו מפריד, אלא בשיטה אחרת: ההודעה פותחת במספר המציין את אורכה, ואחר כך מופיעה התוכן שלה (5 נקודות)

הממשק SpellChecking מגדיר פעולות של בדיקת איות:

```
enum Language {HEBREW, ARABIC, ENGLISH }

interface SpellChecker {
  // returns true if the given word is spelled correctly in the given language
  boolean check(String word, Language lang);

  //returns a list of suggested corrections for the given word in the given language (in case the word is correct, an
  // empty list will be returned)
  List<String> getCorrections(String word, Language lang);
}
```

נגדיר בקשת איות ע"י המחלקה הבא:

```
enum Request  {CHECK, GET_CORRECTIONS }

class SpellMessage {
  final private Request _req;
  final private String word;
  final private Language _lang;

  public SpellMessage(Request  req, String word, Language lang) {
    _req = req; _word = word; _lang = lang;
  }

  Request getRequest() { return _req; }
  String getWord() { return _word; }
  Language getLanguage() { return _lang; }
}
```

הממשק Encoder עודכן, והוא מוגדר כעת לא על בסיס מחרוזת אלא בתבנית כללית:

```
interface Encoder<T> {
  byte[] toBytes(T obj);
  T fromBytes(byte[]);
}
```

ג. השלימו את המחלקה SpellMessgaeEncoder (5 נקודות)

```
class SpellMessageEncoder implements Encoder<SpellMessage> {
 //@TODO
}
```

נתונה מחלקה LexiconBasedSpellChecker, עם בנאי ריק, הממשת באופן שהוא Thread-safe את הממשק SpellChecker. אופן מימושה אינו רלבנטי לשאלה.

ד. עדכנו את השרת, כך שהשרת יבדוק איות של מילים. עליכם להשתמש במחלקות SpellMessageEncoder, MessageLengthBasedTokenizer, SpellMessage (8 נקודות)

נתון מימוש אחר לשרת האיות:

```
class SpellCheckingServer {
  public static void main(String[] args) {
    SpellChecking spellChecker = new LexiconBasedSpellChecker();
    Naming.rebind(args[0],spellChecker);
  }
}
```

ה. עדכנו את הממשק SpellChecker ואת המחלקה LexiconBasedSpellChecker , כדי שהשרת יעבוד (5 נקודות)

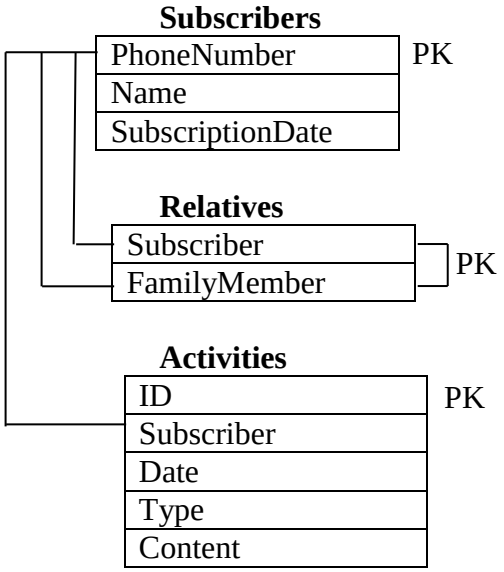
ו. השלימו את תוכנית הלקוח של השרת בסעיף הקודם (2 נקודות):

```
class SpellCheckingClient {
  public static void main(String[] args) {
    SpellChecking spellChecker = null;
    // @TODO
    if (!spellChecker.check("recieve"))
      System.out.println(spellChecker.getCorrections("receive"));
  }
}
```

שאלה 4 (10 נקודות)

במועד א' התוודענו לבסיס הנתונים של חברת הסלולר BigApple המאחסן נתונים על לקוחותיה ועל הפעולות שהם מבצעים במכשיר הנייד שלהם.
עבור כל לקוח נשמרים נתוני שמו, מספר הטלפון שלו, תאריך הצטרפות, וכן קישורים לבני משפחתו, המנויים אף הם בחברת BigApple.
כל פעולה הקשורה ללקוח נשמרת עם תאריך הביצוע שלה. קיימים סוגים שונים של פעולות: שליחת הודעה למספר מסויים, קבלת הודעה ממספר מסויים, התקשרות למספר מסויים, שיחה שלא נענתה ממספר מסויים.

המודל שהוצא בפתרון היה:



א. הרחיבו את מודל הנתונים, כך שיכלול גם את המידע הבא:

- סוג הקשר בין הקרובים (אחים, הורים-ילדים, וכו')
- רשימת חברות הסלולר המתחרות
- רשימת הניודים של הלקוח מהחברה בעבר: תאריך עזיבה, והחברה אליה עבר הלקוח. (5 נקודות)

ב. כתבו שאילתת SQL המחזירה את כל חברות הסלולר שאליהן עבר רינגו סטאר בעבר, ואת תאריכי המעבר. (5 נקודות)

נספח: קוד ה MessagingServer, כפי שנלמד בכיתה ובתרגול

```
public interface Encoder {
	public byte [] toBytes(String s);
	public String fromBytes(byte [] buf);
	public Charset getCharset();
}

public class SimpleEncoder implements Encoder {

	private static final String DFL_CHARSET = "UTF-8";
	private Charset _charset;

	SimpleEncoder() {
		this(DFL_CHARSET);
	}

	SimpleEncoder(String charset) {
		_charset = Charset.forName(charset);
	}

	public byte [] toBytes(String s) {
		return s.getBytes(_charset);
	}

	public String fromBytes(byte [] buf) {
		return new String(buf, 0, buf.length, _charset);
	}

	public Charset getCharset()  {
		return _charset;
	}
}
```

```
public interface Encoder {
	public byte [] toBytes(String s);
	public String fromBytes(byte [] buf);
	public Charset getCharset();
}

public class SimpleEncoder implements Encoder {

	private static final String DFL_CHARSET = "UTF-8";
	private Charset _charset;
```

```
    SimpleEncoder() {
        this(DFL_CHARSET);
    }

    SimpleEncoder(String charset) {
        _charset = Charset.forName(charset);
    }

    public byte [] toBytes(String s) {
        return s.getBytes(_charset);
    }

    public String fromBytes(byte [] buf) {
        return new String(buf, 0, buf.length, _charset);
    }

    public Charset getCharset()  {
        return _charset;
    }
}
```

```
/**
 * This interface care of tokenizing an input stream into protocol specific
 * messages.
 *
 */
public interface Tokenizer {
    /**
     * @return the next token, or null if no token is available. Pay attention
     *      that a null return value does not indicate the stream is closed,
     *      just that there is no message pending.
     * @throws IOException to indicate that the connection is closed.
     */
    String nextToken() throws IOException;

    /**
     * @return whether the input stream is still alive.
     */
    boolean isAlive();

}

public class MessageTokenizer implements Tokenizer {
```

```
    public final char _delimiter;
    private final InputStreamReader _isr;
    private boolean _closed;

    public MessageTokenizer (InputStreamReader isr, char delimiter) {
        _delimiter = delimiter;
        _isr = isr;
        _closed = false;
    }

    public String nextToken() throws IOException {
        if (!isAlive())
            throw new IOException("tokenizer is closed");

        String ans = null;
        try {
            // we are using a blocking stream, so we should always end up
            // with a message, or with an exception indicating an error in
            // the connection.
            int c;
            StringBuilder sb = new StringBuilder();
            // read char by char, until encountering the framing character, or
            // the connection is closed.
            while ((c = _isr.read()) != -1) {
                if (c == _delimiter)
                    break;
                else
                    sb.append((char) c);
            }
            ans = sb.toString();
        } catch (IOException e) {
            _closed = true;
            throw new IOException("Connection is dead");
        }
        return ans;
    }

    public boolean isAlive() {
        return !_closed;
    }

}
```

```
public interface MessagingProtocol {
    /**
     * Process a given message.
     *
     * @return the answer to send back, or null if no answer is required
     */
    String processMessage(String msg);

    /**
    * determine whether the given message is the termination message
    * @param msg the message to examine
    * @return true if the message is the termination message, false otherwise
    */
    boolean isEnd(String msg);

    /**
     * @return true if the connection should be terminated
     */
    boolean shouldClose();

    /**
     * called when the connection was not gracefully shut down.
     */
    void connectionTerminated();

}

public class LinePrintingProtocol implements MessagingProtocol {

    private boolean _shouldClose;
    private int _lineNumber;

    public LinePrintingProtocol() {
        _shouldClose = false;
        _lineNumber = 0;
    }

    public boolean shouldClose() {
        return _shouldClose;
    }

    public void connectionTerminated() {
        _shouldClose = true;
    }

    public String processMessage(String msg) {
```

```
        String ans = null;

        if (msg != null) {
            if (isEnd(msg))
                _shouldClose = true;
            else {
                System.out.println("Message " + _lineNumber + ":" + msg);
                ans = new Date().toString() + ": printed\n";
            }
        }
        return ans;
    }

    public boolean isEnd(String msg) {
        return msg.equalsIgnoreCase("bye");
    }
}
```

```
public class ConnectionHandler implements Runnable {

    private final Socket _socket;
    private final Encoder _encoder;
    private final Tokenizer _tokenizer;
    private final MessagingProtocol _protocol;

    public ConnectionHandler(Socket s, Encoder encoder, Tokenizer tokenizer, MessagingProtocol protocol) {
        _socket = s;
        _encoder = encoder;
        _tokenizer = tokenizer;
        _protocol= protocol;
    }

    public void run() {
        while (!_protocol.shouldClose() && !_socket.isClosed()) {
            try {
                if (!_tokenizer.isAlive())
                    _protocol.connectionTerminated();
                else {
                    String msg = _tokenizer.nextToken();
                    String ans = _protocol.processMessage(msg);
                    if (ans != null) {
                        byte[] buf = _encoder.toBytes(ans);
                        _socket.getOutputStream().write(buf, 0, buf.length);
                    }
```

```
            }
        } catch (IOException e) {
            _protocol.connectionTerminated();
            break;
        }
    }
    try {
        _socket.close();
    } catch (IOException ignored) {
    }
    System.out.println("thread done");
  }
}
```

```
class MessagingServer {
  public static void main(String[] args)
    throws NumberFormatException, IOException
  {
    if (args.length != 1) {
        System.err.println("please supply only one argument, the port to bind.");
        return;
    }

    Encoder encoder = new SimpleEncoder("UTF-8");
    ServerSocket socket = new ServerSocket(Integer.parseInt(args[0]));
    while (true) {
        Socket s = socket.accept();
        Tokenizer tokenizer = new MessageTokenizer(
                new InputStreamReader(s.getInputStream(),encoder.getCharset()),'\n');
        MessagingProtocol protocol = new LinePrintingProtocol();
        Runnable connectionHandler = new ConnectionHandler(s, encoder, tokenizer, protocol);
        new Thread(connectionHandler).start();
    }
  }
}
```