אוניברסיטת בן-גוריון

מדור בחינות

מספר נבחן:_____

רשמו תשובותיכם בגיליון התשובות בלבד
תשובות מחוץ לגיליון לא יבדקו.

בהצלחה!

תאריך הבחינה: 5.3.2012
שם המורה: פרופ' מיכאל אלחדד
ד"ר מני אדלר
ד"ר אנדרי שרף
שם הקורס: תכנות מערכות
מספר הקורס: 202-1-2031
מיועד לתלמידי: מדעי המחשב, הנדסת תוכנה
שנה: תשע"ב
סמסטר: א'
מועד: ב'
משך הבחינה: שלש שעות
חומר עזר: אסור

שאלה 1 (30 נקודות)

בשאלה זו נמשיך לעסוק בממשק **Circle** ובמימושו במחלקה **CircleImpl**.

כזכור, מייצג הממשק **Circle** מעגל במרחב דו ממדי, תחת האילוץ שהקורדינטות של כל הנקודות במעגל הינן חיוביות (במילים אחרות, המעגל ממוקם ברבע החיובי של הצירים X,Y, בו ערכי x ו y גדולים או שווים ל 0). המעגל מיוצג על ידי קורדינטות נקודת המרכז, ואורך הרדיוס.

```
interface Circle {
  //@INV: getRadius() >= 0 && getX() >= getRadius() && getY() >= getRadius()
  int getX();     // returns the x coordinate of the circle center point
  int getY();     // returns the y coordinate of the circle center point
  int getRadius();  // returns the length of the circle radius
  void setCenter(int x, int y)   throws Exception;  // set the coordinates for the circle center point
  void setRadius(int radius)  throws Exception;  // set the length of the circle radius
}
```

להלן קוד המחלקה **CircleImpl** הממשת את הממשק **Circle**, כפי שניתן בפתרון לסעיף ג' בשאלה הראשונה במועד א. המחלקה מסונכרנת סנכרון מלא ובטוחה ונכונה עבור הרצה מקבילית:

```
class CircleImpl implements Circle {
  private int _x;
  private int _y;
  private int _radius;

  CircleImpl(int x, int y, int radius) throws Exception {
   if (!checkInv(x,y,radius))
     throw new Exception();
   _x = x;
   _y = y;
```

```
    _radius = radius;
  }

  public synchronized int getX() {
    return _x;
  }
  public synchronized int getY() {
    return _y;
  }
  public synchronized int getRadius() {
    return _radius;
  }

  public synchronized void setCenter(int x, int y)    throws Exception {
   if (!checkInv(x, y, _radius))
     throw new Exception();
    _x = x;
    _y = y;
  }
  public synchronized void setRadius(int radius)  throws Exception  {
   if (!checkInv(_x, _y, radius))
     throw new Exception();
    _radius = radius;
  }

  protected boolean checkInv(int x, int y, int radius) {
    return radius >= 0 && x >= radius && y >= radius;
  }
}
```

א.לממשק **Circle** נוספה כעת מתודה חדשה

```
void union(Circle other) throws Exception;
```

המתודה מקבלת מעגל אחר (**other**) ומשנה את המעגל הנוכחי (**this**), כך שהוא יכלול את המעגל **other** (כלומר, כל נקודה במעגל **other** תהיה בשטחו של המעגל הנוכחי).

i. הגדירו תנאי התחלה וסיום למתודה **union** [5 נקודות]

נתונה מתודה המחשבת את הרדיוס של מעגל שנקודת המרכז שלו **<x1,y1>**, המאחד את המעגלים **(x1,y1,r1)** ו **(x2,y2,r2)**.
<u>אין צורך להתעמק באופן המימוש.</u>

```
protected int calculateUnionRadius(int x1, int y1, int radius1, int x2, int y2, int radius2) {
```

```
  int left = Math.min( x1 - radius1, x2 - radius2);
  int right = Math.max(x1 + radius1, x2 + radius2);
  int down = Math.min( y1 - radius1, y2 - radius2);
  int up = Math.max(y1 + radius1, y2 + radius2);

  return  Math.max(
    Math.max(x1 - left, right - x1),
    Math.max(y1 - down, up - y1)
  );
}
```

ii. השלימו את מימוש המתודה **union** במחלקה **CircleImpl** (5 נקודות)

```
class CircleImpl implements Circle {
  …
  public synchronized void union(Circle other) throws Exception {
    synchronized (other) {
      //@TODO
    }
  }
  …
}
```

iii. הראו תרחיש בו הרצה מקבילית העושה שימוש ב **CircleImpl** נקלעת למצב של חבק (deadlock) [ניתן לענות על סעיף זה גם אם לא עשיתם את הסעיף הקודם] (4 נקודות)

iv. שנו את הקוד כך שימנע החבק (6 נקודות)

ב. <u>סעיף זה מתייחס לקוד המקורי בתחילת השאלה</u>. <u>ניתן לענות עליו גם אם לא פתרתם את סעיף א</u>.

במימוש הנוכחי לא ניתן לבצע מתודות במקביל. עדכנו את המימוש כך שהוא יאפשר הרצה במקביל של מתודות, על פי המדיניות הבאה:
- ניתן לבצע במקביל פעולות קריאה (getters) ללא הגבלה
- לא ניתן לקרוא כאשר מתבצעת פעולת כתיבה (setters)
- לא ניתן לכתוב כאשר מתבצעת פעולת כתיבה (setters)

(10 נקודות)

<u>חומר עזר</u>: קוד המחלקה Semaphore, כפי שנלמד בכתה

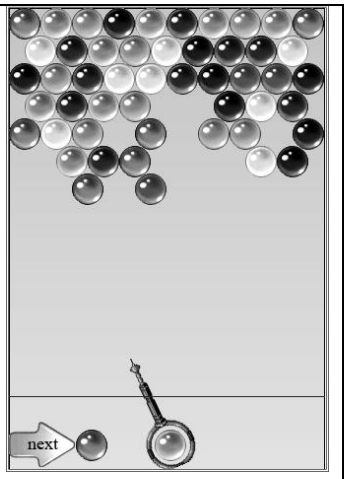
```
class  Semaphore {
   private final int permits_;
   private int free_;
```

```
  public Semaphore(int permits) {
    permits_ = permits;
    free_ = permits;
  }

  // @PRE: free_ > 0 (There is an available permit)
  // @POST: free_ = @pre(free_) - 1
  public synchronized void acquire() throws InterruptedException {
    while (free_<=0){
      this.wait();
    }
    free_ --;
    return;
  }

  // Try acquiring a permit if possible without blocking.
  public synchronized boolean tryAcquire() {
    if (free_ <= 0) return false;
    free_ --;
    return true;
  }

  // @POST: free_ = min(permits_, @pre(free_) + 1)
  public synchronized void release() throws InterruptedException {
    if (free_ < permits_) {
      free_++;
      this.notifyAll();
    }
    return;
  }
}
```

שאלה 2 (30 נקודות)

במשחק "באבלס" המטרה היא לנקות שורות עיגולים מהמסך על-ידי ירי עליהם בעיגולים מאותו הצבע. למפתחי המשחק נודע על הפיתוח המעמיק של מחלקות עיגולים בSPL. עלכן, הם ביקשו לקחת מחלקת Circle בסיסית בC++ ולהרחיבה למחלקת BubblesCircle באופן הבא:

```
class Circle{
public:
        Circle():_x(0),_y(0),_radius(0){;}
        virtual ~Circle(){;}

        int getX() const {return _x;}
        int getY() const {return _y;}
        int getRadius() const {return _radius;}

        void setCenter(int x, int y){_x=x;_y=y;}
        void setRadius(int radius){_radius=radius;}

private:
        int _x;
        int _y;
        int _radius;
};


class BubblesCircle: public Circle{
public:
        BubblesCircle();
        BubblesCircle(int x, int y, int radius, int newradius, const char* color);
        ~BubblesCircle();
        BubblesCircle(const BubblesCircle& b);
        BubblesCircle& operator=(const BubblesCircle& b);

        virtual void setRadius(int p){_new_radius = p;}
        virtual int  getRadius() const {return _new_radius;}

private:
        int   _new_radius;
        char *_color_name;
};
```



א. עזרו למפתחי "באבלס" וממשו את הבנאים, אופרטור השמה ופונקציה הורסת החסרים עבור BubblesCircle. שימו לב כי new_radius_ הוא שדה המחזיק רדיוס נוסף של העיגול לצרכי המשחק, color_name_הוא שם הצבע שבו צבוע אותו עיגול (אין מגבלה על אורך המחרוזת). [10 נקודות]

ב. מפתחי המשחק כתבו את קטע הקוד הבא, ציירו את תמונת הזיכרון המתקבלת מריצה עד ההערה: [10 נקודות]

```
BubblesCircle dummy(Circle &c1, Circle *c2){
   c1.setRadius(c2->getRadius());
   //***here***
```

```
    return c1;
}

void main(){
    char *color = "blue";
    BubblesCircle c1(1,2,3,4, color);
    Circle *c2 = new Circle();
    BubblesCircle c3 = dummy(c1, c2);
}
```

ג. כתבו פונקציה המקבלת וקטור של עיגולים ושני צבעים, צבע מקור (s) וצבע יעד (t). הפונקציה מחליפה את כל העיגולים בצבע (_color_name) s לצבע החדש t: [10 נקודות]

```
void ReplaceColor(vector<BubblesCircle> &set, const char *s, const char *t);
```

שאלה 3 (30 נקודות)

בנספח למבחן מופיע קוד ה Reactor הממש שרת חיפוש, כפי שהופיע בפתרון למועד א.
כזכור, ממש השרת פרוטוקול המקבל מהלקוח הודעה המכילה מילה לחיפוש ומחזיר ללקוח את רשימת הקבצים בהם היא מופיעה.
השרת משתמש במחלקה **SearchService** המכילה את המתודה **search(word)**. מתודה זו מקבלת מילה לחיפוש ומחזירה רשימה של שמות קבצים בהם המילה מופיעה. המחלקה עושה שימוש בספריית קבצי אינדקס, כאשר כל קובץ אינדקס מכיל מילים המתחילות באות מסוימת, כאשרלכל מילה ניתנת רשימה של שמות מסמכים בהם מופיע המילה.
<u>קוד השרת, כי שניתן בפתרון סעיף ב בשאלה 3 במועד א מופיע בנספח למבחן. מדובר בגרסה הפשוטה של השרת בה מתבצעת הקריאה מקבצי האינדקס ב blocking io</u>.

במימוש הנוכחי חילוץ רשימת המסמכים מקבצי האינדקס, עבור מילה נתונה לחיפוש מתבצע כחלק מתהליך שרת ה Reactor. עיצוב שכזה עשוי להיות בעייתי עבור שרתי חיפוש דוגמת גוגל, שכן פעולת החיפוש בקבצי האינדקס הינה כבידה.
בשאלה זו נתמודד עם בעיה זו על ידי הגדרת שירות החיפוש בקובץ אינדקס אחד כ Remote Object הרץ על host משלו, בתבנית העיצוב הבאה:

- לכל קובץ אינדקס מוגדר Remote Object התומך בחילוץ רשימת המסמכים בהם מופיעה מילה נתונה.
- שרת ה Reactor פונה ל rmiregistry לשם קבלת גישה ל Remote Objects השונים של קבצי האינדקס.
- שרת ה Reactor ממש את פרוטוקול החיפוש על ידי פניה ל Remote Object האחראי על אות ההתחלה של המילה לחיפוש.

בסעיפים הבאים נממש את התבנית המוצעת:

א. הגדירו את הממשק IndexService הממש Remote וכן פעולת חילוץ רשימה של מסמכים מקובץ אינדקס, עבור מילה נתונה [5 נקודות]

ב. כתבו את המחלקה IndexServiceImpl הממשת את הממשק IndexService (ניתן לקרוא באופן פשוט את תוכן קובץ האינדקס ב blocking io) [5 נקודות]

ג. השלימו את המחלקה IndexServer, המייצרת אובייקט מסוג IndexService, ומפרסמת אותו ב rmiregistry [5 נקודות]

```
class IndexServer {
  public static void main(String[] args) {
    String rmiHost = args[0];
    int rmiPort = Integer.parseInt(args[1]);
    //@TODO
  }
}
```

ד. עדכנו את קוד ה Reactor כך שחילוץ רשימת המסמכים מהאינדקס המתאים למילה הנתונה לחיפוש, יתבצע בעזרת ה Remote Object המתאים [10 נקודות]

ה. תארו בקצרה את השלבים להרצת המערכת כולה – אילו תהליכים יש להריץ ובאיזה סדר [5 נקודות]

שאלה 4 (10 נקודות)

במועד א הגדרנו מודל נתונים עבור קטלוג הסרטים של מועדון וִשֶׁרֶט לָנֶפֶש, המכיל את המידע הבא:
<u>סרטים</u>: שם הסרט, ארץ ושנת הוצאה, שם הבמאי
<u>במאים</u>: שם הבמאי, רשימת סרטים
<u>שחקנים</u>: שם השחקן, רשימת תפקידים בסרטים

להלן מודל הנתונים, כפי שניתן בפתרון מועד א:

```
Create table movies(
   Int id primary key;
   Varchar(200) name;
   Varchar(80) country;
   Date publicationDate;
   Int directorId foreign key references directors(id);
 )

Create table directors (
   Int id primary key;
   Varchar(80) name;
)

Create table actors (
   Int id primary key;
   Varchar(80) name;
)

Create table actors_movies (   -- cross table
   Int actorId foreign key references actors(id);
```

```
    Int movieId foreign key references movies(id);
    Varchar(80) role;
    Primary key (actorId, movieId);
)
```

א. במודל הנוכחי שחקן מופיע רק בתפקיד אחד בכל סרט. כדי לתמוך בסרטים דוגמת 'חגיגה בסנוקר' ו'שלמה המלך ושלמי הסנדלר', נדרש כעת לעדכן את המודל כך ששחקן עשוי להופיע במגוון תפקידים באותו סרט.
שנו את המודל כך שיתמוך בדרישה זו [5 נקודות]

ב. כתבו שאילתת SQL המחזירה את רשימת התפקידים שמבצע חיים טופול בסרטים שביים אפרים קישון [5 נקודות]

נספח: החלק העיקרי של קוד Reactor המממש שרת חיפוש, כפי שניתן בפתרון לשאלה 3 סעיף ב במועד א.

```
public class Reactor implements Runnable {
   private static final Logger logger = Logger.getLogger("edu.spl.reactor");
   private final int _port;
   private final int _poolSize;
   private final ServerProtocolFactory _protocolFactory;
   private final TokenizerFactory _tokenizerFactory;
   private volatile boolean _shouldRun = true;
   private ReactorData _data;

   public Reactor(int port, int poolSize, ServerProtocolFactory protocol, TokenizerFactory tokenizer) {
      _port = port;
      _poolSize = poolSize;
      _protocolFactory = protocol;
      _tokenizerFactory = tokenizer;
   }
  …
   public void run() {
      ExecutorService executor = Executors.newFixedThreadPool(_poolSize);
      Selector selector = null;
      ServerSocketChannel ssChannel = null;
      try {
         selector = Selector.open();
         ssChannel = createServerSocket(_port);
      } catch (IOException e) {
         logger.info("cannot create the selector -- server socket is busy?");
         return;
      }
      _data = new ReactorData(executor, selector, _protocolFactory, _tokenizerFactory);

      ConnectionAcceptor connectionAcceptor = new ConnectionAcceptor( ssChannel, _data);
      try {
         ssChannel.register(selector, SelectionKey.OP_ACCEPT, connectionAcceptor);
      } catch (ClosedChannelException e) {
         logger.info("server channel seems to be closed!");
         return;
      }

      while (_shouldRun && selector.isOpen()) {
         try {
            selector.select();
         } catch (IOException e) {
```

```
            logger.info("trouble with selector: " + e.getMessage());
            continue;
        }
        Iterator it = selector.selectedKeys().iterator();
        while (it.hasNext()) {
            SelectionKey selKey = (SelectionKey) it.next();
            it.remove();

            if (selKey.isValid() && selKey.isAcceptable()) {
                logger.info("Accepting a connection");
                ConnectionAcceptor acceptor = (ConnectionAcceptor) selKey.attachment();
                try {
                    acceptor.accept();
                } catch (IOException e) {
                    logger.info("problem accepting a new connection: "
                            + e.getMessage());
                }
                continue;
            }
            if (selKey.isValid() && selKey.isReadable()) {
                ConnectionHandler handler = (ConnectionHandler) selKey.attachment();
                logger.info("Channel is ready for reading");
                handler.read();
            }
            if (selKey.isValid() && selKey.isWritable()) {
                ConnectionHandler handler = (ConnectionHandler) selKey.attachment();
                logger.info("Channel is ready for writing");
                handler.write();
            }
        } // All fired events have been processed
    } // Either server asked to shutdown the reactor or selector "broke" on an exception.
    stopReactor();
}
...
public static void main(String args[]) {
    if (args.length != 2) {
        System.err.println("Usage: java Reactor <port> <pool_size>");
        System.exit(1);
    }
    try {
        ServerProtocolFactory protocolMaker = new ServerProtocolFactory() {
            public AsyncServerProtocol create() {
              return new SearchProtocol()
             }
```

```
        };
        final Charset charset = Charset.forName("UTF-8");
        TokenizerFactory tokenizerMaker = new TokenizerFactory() {
          public StringMessageTokenizer create() {
            return new FixedSeparatorMessageTokenizer("\n", charset);
          }
        };
        int port = Integer.parseInt(args[0]);
        int poolSize = Integer.parseInt(args[1]);
        Reactor reactor = new Reactor(port, poolSize, protocolMaker, tokenizerMaker);
        Thread thread = new Thread(reactor);
        thread.start();
        logger.info("Reactor is ready on port " + reactor.getPort());
        thread.join();
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
}
```

```
public class ConnectionAcceptor {
    private static final Logger logger = Logger.getLogger("edu.spl.reactor");
    protected ServerSocketChannel _ssChannel;
    protected ReactorData _data;

    public ConnectionAcceptor(ServerSocketChannel ssChannel, ReactorData data) {
        _ssChannel = ssChannel;
        _data = data;
    }

    public void accept() throws IOException {
        SocketChannel sChannel = _ssChannel.accept();
        if (sChannel != null) {
            SocketAddress address = sChannel.socket().getRemoteSocketAddress();

            logger.info("Accepting connection from " + address);
            sChannel.configureBlocking(false);
            SelectionKey key = sChannel.register(_data.getSelector(), 0);

            ConnectionHandler handler = ConnectionHandler.create(sChannel, _data, key);
            handler.switchToReadOnlyMode(); // set the handler to read only mode
        }
    }
}
```

```
public class ConnectionHandler {
  protected final SocketChannel _sChannel;
  protected final ReactorData _data;
  protected final AsyncServerProtocol _protocol;
  protected final StringMessageTokenizer _tokenizer;
  protected Vector<ByteBuffer> _outData;
  protected final SelectionKey _skey;
  private ProtocolTask _task;
  …
  public synchronized void addOutData(ByteBuffer buf) {
    _outData.add(buf);
    switchToReadWriteMode();
  }

  public void read() {
    if (_protocol.shouldClose())
      return;
    SocketAddress address = _sChannel.socket().getRemoteSocketAddress();
    logger.info("Reading from " + address);
    ByteBuffer buf = ByteBuffer.allocate(BUFFER_SIZE);
    int numBytesRead = 0;
    try {
      numBytesRead = _sChannel.read(buf);
    } catch (IOException e) {
      numBytesRead = -1;
    }
    if (numBytesRead == -1) {      // Is the channel closed?
      logger.info("client on " + address + " has disconnected");
      closeConnection();
     _protocol.connectionTerminated();
      return;
    }
    buf.flip();
    _task.addBytes(buf);
    _data.getExecutor().execute(_task);
  }

  public synchronized void write() {
    if (_outData.size() == 0) { // if nothing left in the output string, go back to read mode
      switchToReadOnlyMode();
      return;
    }
    ByteBuffer buf = _outData.remove(0);
    if (buf.remaining() != 0) {
```

```
        _sChannel.write(buf);
        if (buf.remaining() != 0)
          _outData.add(0, buf);
      }
      if (_protocol.shouldClose()) {
        switchToWriteOnlyMode();
        if (buf.remaining() == 0) {
          logger.info("disconnecting client on " + _sChannel.socket().getRemoteSocketAddress());
          closeConnection();
        }
      }
    }
...
}
```

```
public class ProtocolTask implements Runnable {
    private final ServerProtocol _protocol;
    private final StringMessageTokenizer _tokenizer;
    private final ConnectionHandler _handler;
    private final Vector<ByteBuffer> _buffers = new Vector<ByteBuffer>();

    public ProtocolTask(final ServerProtocol protocol, final StringMessageTokenizer tokenizer,
                        final ConnectionHandler h) {
        this._protocol = protocol;
        this._tokenizer = tokenizer;
        this._handler = h;
    }

    public synchronized void run() {
          synchronized (_buffers) {
              while(_buffers.size() > 0) {
                  ByteBuffer buf = _buffers.remove(0);
                  this._tokenizer.addBytes(buf);
              }
          }
          while (_tokenizer.hasMessage()) {
              String msg = _tokenizer.nextMessage();
              String response = this._protocol.processMessage(msg);
              if (response != null) {
                  try {
                      ByteBuffer bytes = _tokenizer.getBytesForMessage(response);
                      this._handler.addOutData(bytes);
                  } catch (CharacterCodingException e) { e.printStackTrace(); }
              }
```

```
        }
    }

    public void addBytes(ByteBuffer b) {
        // We synchronize on _buffers and not on "this" because
        // run() is synchronized on "this", and it might take a long time
        // to run.
        synchronized (_buffers) {
      _buffers.add(b);
        }
    }
}
```

```
public class SearchProtocol implements AsyncServerProtocol {

    SearchService _s = new SearchService();

    public boolean isSearch(String msg) {
      return msg.equals("search");
    }

    // Assume the command comes as “search <queryWord>”
    public String getQuery(String searchCmd) {
       String[] ws = searchCmd.split(“ “);
       return ws[1];
    }

    // (‘a’, ‘b’) → ‘a#b#’
    public String listToString(List<String> ls) {
       StringBuilder ret = new StringBuilder();
       for (String s : ls) {
          ret.append(s);
          ret.append(‘#’);
       }
       return ret.toString();
    }

    public String processMessage(String msg) {
      if (this._connectionTerminated) {
        return null;
      }
      if (this.isEnd(msg)) {
        this._shouldClose = true;
        return "Ok, bye bye";
      } else if (this.isSearch(msg)) {
          String query = this.getQuery(msg);
          List<String> ls = _s.search(query);
                return this.listToString(ls);
```

```
            } else {
                return "Unknown command";
            }
    }
}
```

```
class SearchService {
  private Map<Character,File> _indexFiles;
  …
  public List<String> search(String word) {
    try {
      List<Byte> bytes = new LinkedList<Byte>();
      File index = _indexFiles.get(word.charAt(0));
      FileInputStream in = new FileInputStream(index);
     int b = -1;
      while ((b = in.read()) != -1)
        bytes.add((byte)b);
      return search1(word, bytes);
    } catch (Exception e) {
      return new LinkedList<String>();
    }
  }

  protected List<String> search1(String word, List<Byte> bytes) {
    // finds the given word in the given list of bytes, and returns the data (=a list of file names which contains
    // this  word) attached to this word
    …
  }

  protected List<String> search2(String word, List<ByteBuffer> bytes) {
   // finds the given word in the given list of ByteBuffers, and returns the data (=a list of file names which contains
   //  this  word) attached to this word
    …
  }
}
```