אוניברסיטת בן-גוריון

מדור בחינות

מספר נבחן:_____

רשמו תשובותיכם בגיליון התשובות בלבד
תשובות מחוץ לגיליון לא יבדקו.

בהצלחה!

תאריך הבחינה: 3.3.2006
שם המורה: ד"ר מיכאל אלחדד
מני אדלר
שם הקורס: תכנות מערכות
מספר הקורס: 202-1-2031
מיועד לתלמידי: מדעי המחשב, הנדסת תוכנה
שנה: תשס"ו
סמסטר: א'
מועד: ב'
משך הבחינה: שלש שעות
חומר עזר: אסור

שאלה 1 (30 נקודות)

להלן הקוד של התוכנית שהוצגה בשאלה הראשונה במועד א' (הקוד זהה לחלוטין, למעט שורת קוד נוספת, שאינה רלבנטית לשאלה זו, הבודקת את תקינות הנתונים שנשלחו לבונה של BankAccount).
תזכורת קצרה לאילו מביניכם שהספיקו לשכוח:

- המחלקה Stock מייצגת מניה של חברה בשוק ההון. מניה מורכבת משם (_name), שער בסיסי (_base), מספר קונים (_ref), ומחיר (_price). מחיר המניה הוא סכום השער הבסיסי ומכפלת מספר הקונים ב 0.1.
- המחלקה BankAccount מייצגת חשבון של לקוח בבנק, החשבון מורכב ממספר חשבון (_ID), סכום הכסף בחשבון עו"ש (_savings), וטבלת המניות שבבעלות בעל החשבון (_stocks). הטבלה ממפה מניה, למספר היחידות שנקנו על ידי הלקוח. הבנק מאפשר משיכת יתר (אוברדרפט) שגובהה מוגבל על ידי השדה _maxOverDraft.
- המחלקה Dealer מייצגת אובייקט אקטיבי המכיל חשבונות בנק ומניות, ותור של הוראות לביצוע (אובייקטים המממשים את הממשק Command), ומבצע את ההוראות על חשבונות הבנק והמניות, על ידי הפעלת מתודת apply של Command.
- קיימים רק המימושים BuyCommand , SellCommand למשק Command.
- המחלקה SynchRandomCollection הינה מבנה נתונים הממש באופן מסונכרן ותחת מנגנון wait/notify פעולות הוספה והסרה של איברים מאוסף, כאשר פעולת add(Object) מוסיפה איבר לאוסף, ופעולת remove() מסירה איבר אקראי כלשהוא מהאוסף. כמו כן המחלקהSynchMap הינה מימוש של Map כך שכל הפעולות עליה, ובפרט המתודה get(Object), מסונכרות.
- נתון תהליך בו רצים שני Dealers מעל אותה קבוצת חשבונות, אותה קבוצת מניות, ואותה קבוצת פקודות.

```
class Stock  {
        private final String name_;
        final private double base_;
        private int ref_;
        private double price_;

        //@INV:  base_ >= 0 && ref_ >= 0 && price_ ==  (base_ + 0.1 * ref_)
        Stock(String name,double base) throws WrongStockBasePriceException {
          if (base < 0)
```

```
            throw new WrongStockBasePriceException(base);
          base_ = base;              ref_ = 0;
          price_ = base;             name_ = name;
        }
        public synchronized double getPrice() { return price_; }
        public synchronized void incRef() { ref_++;  resetPrice(); }
        public synchronized void decRef() {
          if (ref_ > 0) {
            ref_--;
            resetPrice();
          }
        }

        private synchronized void resetPrice() {
          price_ = base_ + 0.1 * ref_;
        }

        public boolean equals(Object o) {
          return name_.equals(((Stock)o).name_);
        }
      }
```

```
class BankAccount {
        private final int ID_;
        private double savings_;
        private double maxOverDraft_;
        private HashMap<Stock,Integer> stocks_;
        //@INV: savings_ + maxOverDraft_ >= 0

        BankAccount(int ID, double amount, double maxOverDraft)
         throws WrongAccountDataException {
          if (ID  < 1 || maxOverDraft < 0 || amount < maxOverDraft )
            throw new WrongAccountDataException();
          ID_ = ID;     savings_ = amount;   maxOverDraft_ = maxOverDraft;
          stocks_ =  new HashMap<Stock,Integer>();
        }
        public synchronized void incSaving(double amount) { savings_ += amount; }
        public synchronized void decSaving(double amount) { savings_ -= amount; }
        public synchronized void buyStock(Stock stock) {
                if ((savings_ + maxOverDraft_)  >= stock.getPrice()) {
```

```
                        Integer num = stocks_.get(stock);
                        if (num == null)
                           stocks_.put(stock,new Integer(1));
                        else
                           stocks_.put(stock,new Integer(num.intValue() +1));
                        decSaving(stock.getPrice());
                        stock.incRef();
                }
        }

        public synchronized void sellStock(Stock stock) {
                Integer num = stocks_.get(stock);
                if (num != null) {
                        if (num.intValue() > 1)
                          stocks_.put(stock, num.intValue() -1);
                        else
                          stocks_.remove(stock);
                        incSaving(stock.getPrice());
                        stock.decRef();
                }
        }
}
```

```
interface Command {
   public void apply(SynchMap<Integer,BankAccount> accounts,
                        SynchMap<String,Stock> stocks);
}

class Dealer extends Thread {

        SynchMap<String,Stock> stocks_;
        SynchMap<Integer,BankAccount> accounts_;
        SynchRandomCollection <Command> commands_;

        Dealer(SynchMap<Integer,BankAccount> accounts,
                SynchMap<String,Stock> stocks,
                SynchRandomCollection<Command> commands)
          { stocks_ = stocks;  accounts_ = accounts;  commands_ = commands; }

        public void run() {
          while (true) {
```

```
            Command command  = commands_. remove();
            command.apply(accounts_,stocks_);
          }
        }
      }
```

```
class BuyCommand implements Command  {
      protected Integer accountID_;
      protected String stockName_;
      BuyCommand(int accountID,String stockName) {
                accountID_ = new Integer(accountID);
                stockName_ = stockName;
      }

      public void apply(SynchMap<Integer,BankAccount> accounts ,
                              SynchMap<String,Stock> stocks)    {
                BankAccount account = accounts.get(accountID_);
                Stock stock = stocks.get(stockName_);
                if (account != null &&  stock  != null)
                  account.buyStock(stock);
      }
}
```

```
class SellCommand  implements Command {
      protected Integer accountID_;
      protected String stockName_;
      SellCommand(int accountID,String stockName)  {
                accountID_ = new Integer(accountID);
                stockName_ = stockName;
      }
      public void apply(SynchMap<Integer,BankAccount> accounts ,
                              SynchMap<String,Stock> stocks)    {
                BankAccount account = accounts.get(accountID_);
                Stock stock = stocks.get(stockName_);
                if (account != null &&  stock  != null)
                  account.sellStock(stock);
      }
}
```

א. במועד א', נוכחנו לדעת כי המחלקה BankAccount אינה בטוחה, וכן כי הרצת התוכנית אינה נכונה. אחד הסטודנטים הציע לפתור את בעיית הבטיחות והנכונות על ידי שכתוב המתודה buyStock באופן הבא:

```
public synchronized void buyStock(Stock stock) {
  int price = stock.getPrice();
  if ((savings_ + maxOverDraft_)  >= price) {
    Integer num = stocks_.get(stock);
    if (num == null)
      stocks_.put(stock,new Integer(1));
    else
      stocks_.put(stock,new Integer(num.intValue() +1));
    decSaving(price);
    stock.incRef();
  }
}
```

א1. האם כעת המחלקה BankAccount בטוחה? (6 נקודות)
א2. האם כעת הרצת התהליך הנ"ל (על קבוצת פקודות נתונה) נכונה מבחינת התאמת כל ביצוע מקבילי אפשרי לביצוע סדרתי, על סדר פעולות כלשהוא של קבוצת הפקודות. אם כן הסבירו למה, אם לא הביאו דוגמא נגדית (6 נקודות)

הערה: ניתן להניח, לשם נוחות, שסעיפים ב-ג מתייחסים לקוד המקורי בתחילת השאלה ולא לקוד המתוקן בסעיף א'.

ב. הסבירו מדוע המחלקה BankAccount אינה בטוחה אף במודל חישוב סדרתי, אם נשנה את האינווריאנטה ל:

```
//@INV:  savings_ + stocksValue() + maxOverDraft_  >= 0
```

כאשר stocksValue() הינה מתודה של BankAccount המחזירה את ערך המניות בחשבון (6 נקודות)

```
public synchronized double stocksValue() {
  double ret = 0;
  for (Entry<Stock,Integer> entry : stocks_entrySet())  {
    Stock stock = entry.getKey();
    Integer amount = entry.getValue();
    ret += (amount.intValue() * stock.getPrice());
  }
  return ret;
}
```

ג. בקוד הנתון של buyStock(), אם ללקוח אין מספיק כסף, פעולת הרכישה לא תתבצע. שנו את הקוד של המחלקה BankAccount כך שהליך הקניה יושהה עד שהקניה תתאפשר, והראו כיצד גישה נאיווית זו יכולה להביא את הרצת התהליך עם שני ה Dealers לחבק (deadlock). (12 נקודות)

שאלה 2 (10 נקודות)

נתונה המחלקה Polygon:

```
class Polygon {
public:
  Polygon() : area_(-1) {}
  void addPoint(Point& pt ) {
    invalidateArea();
    points_.push_back(pt);
  }

  Point& getPoint(const int I )   {  return points_[i];  }
  int getNumPoints()  { return points_.size(); }
  double getArea() {
    if( area_ < 0 ) // if not yet calculated and cached
       calcArea();     // calculate now
     return area_;
  }
private:
  void invalidateArea()  { area_ = -1; }
  void calcArea() {
    area_ = 0;
    vector<Point>::iterator it;
    for( it = points_.begin(); it != points_.end(); ++it )
      area_ += /* some work */;
    }
    vector<Point> points_;
    double   area_;
  };
```

עימדו על הבעיות בהגדרת המחלקה מבחינת ה const ותקנו אותה בהתאם.

שאלה 3 (10 נקודות)

נתונה ההגדרה של מחלקה X בקובץ ה header שלה:

```
// x.h: original header
#include <iostream>
#include <list>
// None of A, B, C, D or E are templates.
// Only A and C have virtual functions.
#include "a.h" // class A
#include "b.h" // class B
#include "c.h" // class C
#include "d.h" // class D
#include "e.h" // class E

class X : public A, private B
{
public:
  X( const C& );
  B f( int, char* );
  C f( int, C );
  C& g( B );
  E h( E );
private:
  std::list<C> clist_;
  D d_;
};
```

הגדירו מחדש את המחלקה, עם מינימום תלויות של הקובץ במודולים אחרים, כך שישאר אך ורק #include "a.h".

שאלה 4 (10 נקודות)

נתונה הגדרתה של המחלקה Point, המייצגת נקודה במרחב דו ממדי:

```
class Point  {
public:
  Point(int x,int y) { x_ = new int(x));  y_ = new int(y); }
  Point(const Point& other) { x_ = new int(*other.x_); y_ = new int(*other.y_); }
  ~Point() { delete x_; delete y_; }
  int getX() const { return *x_; }
  int getY() const { return *y_; }
protected:
  const int* x_;
```

```
  const int* y_;
};
```

ממשו את האופרטורים הבאים (אין לשנות את הגדרת המחלקה)

```
Point& operator=(const Point& other);
bool operator==(const Point& other);
Point& operator+=( const Point& other);
```

שאלה 5 (10 נקודות)

ציינו נכון (V) או לא נכון (X) עבור הקביעות הבאות, המתייחסות ל RMI:

א. הפעלת מתודה של RemoteObject אינה תלויה בהכרח בקיומו של תהליך rmiregistry כלשהוא.
ב. תקשורת בין תהליכים עם Remote Objects בטוחה יותר – מבחינת קבלת ההודעות שנשלחו - מתקשורת בעזרת sockets בפרוטוקול TCP.
ג. על תהליך בו מוגדר RemoteObject להכיר את כל הממשקים של הפרמטרים הנשלחים ל RemoteObject.
ד. שליחת RemoteObject כפרמטר למתודה של RemoteObject אחר גוררת נעילה שלו בתהליך המקורי בו הוא הוגדר.
ה. הבחירה בשימוש ב RMI לכתיבת שרת, תלויה באופן מימוש הלקוח.

שאלה 6 (20 נקודות)

נתון שרת (server) הממומש בתבנית ה Reactor.
בישיבת הפיתוח האחרונה, ציין נציג מחלקת שרות הלקוחות, כי לקוחות (המשתמשים בתהליך של client לתהליך ה server) ממתינים זמן רב להתחברות הראשונית לשרת. בישיבה הוחלט לנסות לשפר גם את השרת, כך שתהליך ההתחברות יהיה מהיר יותר, והועלו שלוש עמדות:

I מאחר ש ServerSocketChannel המקבל את בקשות ההתחברות הוא Non Blocking, ולאור העובדה שכל אירוע OP_ACCEPT מזוהה על ידי ה Selector וגורר הפעלת המתודה ()accept שלServerSocketChannel ההצעות האחרות לשיפור (II,III) לא משמעותיות.

II נגדיר עשרה Non-Blocking ServerSocketChannels בת'רד של ה Reactor, כאשר כל אחד מהם מאזין ל port משלו המושם ב Selector עבור ארוע OP_ACCEPT עם ConnectionAcceptor משלו. תהליך הלקוח יכול לפנות לכל אחד מ ports אלו לשם בקשת התחברות. בהגיע בקשת התחברות לאחד מה ServerSocketChannels היא תטופל על ידי הת'רד של ה Reactor באותו אופן שבו זה נעשה עד היום עבור ServerSocketChannel אחד.

III כמו הצעה II אלא שבשונה מההצעה הקודמת, בהגיע בקשת התחברות לאחד מה SereerSocketChannels יועבר ה ConnectionAcceptor לתור המשימות של ה Executor. בהצעה זו ConnectionAcceptor מממש את המתודה ()run על ידי קריאה למתודה ()accept.

א. איזו מהעמדות הנ"ל תקבל/י? נמקו (10 נקודות)
ב. ממשו את ההצעה השלישית. לרשותכם קטעים נבחרים מהקוד הנוכחי (10 נקודות)

```
public class Reactor extends Thread {
```

```
  protected Vector<Integer> _ports;  // vector of 10 port for the 10 ServerSocketsChannels
  protected ExecutorService _pool;
  protected boolean _shouldRun;
  ...
  public void run() {
    try {
      Selector selector = Selector.open();
      ServerSocketChannel ssChannel = ServerSocketChannel.open();
      ssChannel.configureBlocking(false);
      ssChannel.socket().bind(new InetSocketAddress(_ports.get(0));
      ssChannel.register(selector, SelectionKey.OP_ACCEPT,
                                    new ConnectionAcceptor(selector, ssChannel, _pool));
      while (_shouldRun) {
        selector.select();
        Iterator it = selector.selectedKeys().iterator();
        while (it.hasNext()) {
          SelectionKey selKey = (SelectionKey)it.next();
          it.remove();
          if (selKey.isValid() && selKey.isAcceptable()) {
            ConnectionAcceptor connectionAcceptor = (ConnectionAcceptor)selKey.attachment();
            connectionAcceptor.accept();
          }
          if (selKey.isValid() && selKey.isReadable()) {
            ConnectionReader connectionReader = (ConnectionReader)selKey.attachment();
            connectionReader.read();
          }
        }
      }
    } catch (IOException e) {
      e.printStackTrace(System.err);
      stopReactor();
    }
    stopReactor();
  }
  ...
}
```

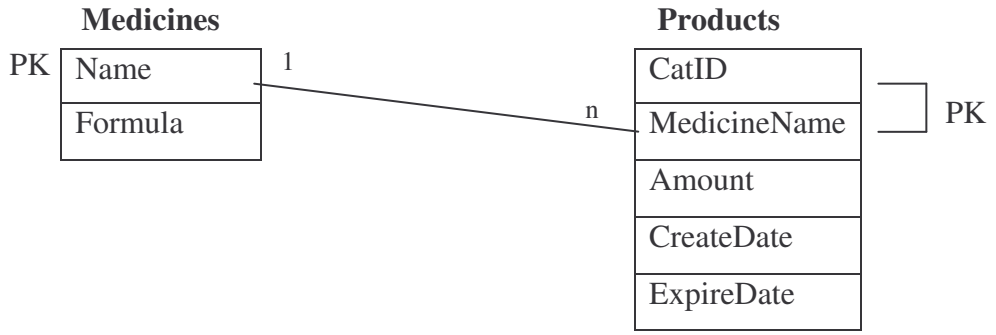
```
public class ConnectionAcceptor {
  protected Selector _selector;
```

```
    protected ServerSocketChannel _ssChannel;
    protected ExecutorService _pool;
    ...
    public void accept() throws IOException {
        // Get a new channel for the connection request
        SocketChannel sChannel = _ssChannel.accept();

        // If serverSocketChannel is non-blocking, sChannel may be null
        if (sChannel != null) {
            SocketAddress address = sChannel.socket().getRemoteSocketAddress();
            System.out.println("Accepting connection from " + address);
            sChannel.configureBlocking(false);
            sChannel.register(_selector, SelectionKey.OP_READ,
                                        new ConnectionReader(sChannel, _pool));
        }
    }
}
```

שאלה 7 (10 נקודות)

במועד א' בנינו מודל הנתונים עבור חברת התרופות, להזכירכם:
- תרופה מוגדרת על ידי שם, ונוסחת התרכובת של התרופה.
- בכל פעם שמופעל קו הייצור של תרופה, ניתן מספר קטלוגי לקבוצת התרופות שיוצרה בפעם זו, תוך ציון מספר העותקים שיוצרו, ותאריך הייצור ותאריך התפוגה של קבוצה זו.



כעת נודע כי תהליך ייצור התרופות מתבצע באחד ממפעלי החברה, כאשר מפעל מוגדר על ידי מספר סידורי, שם, ומקום.

א. עדכנו את מודל הנתונים כך שיכיל את מפעלי הייצור, ויכלול עבור כל קבוצת תרופות שיוצרה ביום מסוים, גם את המפעל בו היא יוצרה. זכרו כי על המודל להיות שלם ומינימאלי. (5 נקודות)
ב. כתבו שאילתת SQL המחזירה את שמות התרופות ונוסחתן שיוצרו אי פעם במפעל בירוחם. (5 נקודות)