# אוניברסיטת בן-גוריון

## מדור בחינות

**מספר נבחן:**_____

רשמו תשובותיכם בגיליון התשובות בלבד
תשובות מחוץ לגיליון לא יבדקו.

**בהצלחה!**

תאריך הבחינה: 24.3.2009
שם המורה: ד"ר מיכאל אלחדד
ד"ר מני אדלר
מר אוריאל ברגיג
שם הקורס: תכנות מערכות
מספר הקורס: 202-1-2031
מיועד לתלמידי: מדעי המחשב, הנדסת תוכנה
שנה: תשס"ט
סמסטר: א'
מועד: א'
משך הבחינה: שלש שעות
חומר עזר: אסור

## שאלה 1 (30 נקודות)

משחק הקלפים הינו משחק מרתק, המשלב הנאה, זריזות ידיים, בהירות מחשבה, וקורט של מזל (ראו איור מספר 1). החוקרים חלוקים בדבר מוצאו של המשחק. יש המייחסים אותו לסינים, אחרים טוענים כי צוענים הביאו אותו לאירופה מהודו, ויש אף הסוברים שמוצאו בארצות ערב, משם הובא לספרד על ידי סוחרים וצלבנים חוזרים.
בשאלה זו נעסוק במימוש גרסאות שונות של המשחק כתוכנית Java, לא נדרש כל ידע מוקדם על המשחק וחוקיו (כמו גם התעמקות בנאמר למעלה).

*איור מספר 1: הקלפנים (סזאן, 92-1890)*

המשחק בו נעסוק מוגדר באופן הבא:

- הקלפים במשחק מאופיינים על ידי מספר שלם בתחום [1-10].
- במשחק משתתפים שני מתחרים.
- כל אחד משני המתחרים מקבל חפיסת קלפים עם 32 קלפים אקראיים.
- לכל מתחרה יש מונה של 'נקודות חובה' המאותחל לאפס.
- על השולחן מונחת ערימה של קלפים. בתחילת המשחק מכילה ערימה זו קלף אחד עם ערך 5.
- מכאן ואילך לוקח כל מתחרה, בקצב שלו (כלומר, המשחק לא מתנהל תור-תור), את הקלף העליון מחפיסתו, ומניח אותו על הקלף בראש הערימה בשולחן. ההפרש המוחלט, בין ערך הקלף המונח על הלוח בשולחן לבין זה של המתחרה, מתווסף לנקודות החובה שלו.
- המשחק מסתיים כאשר הונחו כל קלפי השחקנים על הלוח.
- מנצח המשחק הינו השחקן בעל מספר נקודות החובה הנמוך יותר.

דוגמא של מספר מהלכים במשחק:

| | |
|---|---|
| Table: [5] | // Initial state – players have only 3 cards |
| Player 1: [9,7,9] | Debt = 0 |
| Player 2: [6,8,1] | Debt = 0 |
| | |
| Table: [5,9,7] | // Player 1 puts down cards 9, 7 and Player 2 does nothing |
| Player 1: [9] | Debt = \|5-9\| + \|9-7\| = 6 |
| Player 2: [6,8,1] | Debt = 0 |
| | |
| Table: [5,9,7,6,8,9] | // Player 2 puts down cards 6,8 and Player 1 puts down 9 |
| Player 1: [] | Debt = 6 + \|9-8\| = 7 |
| Player 2: [1] | Debt = \|6-7\| + \|8-6\| = 3 |

להלן מימוש סימולציה של המשחק: האובייקטים האקטיביים הינם שני השחקנים (Player), והאובייקט הפסיבי המשותף הינו שולחן המשחק (CardTable).
בקוד נעשה שימוש בממשקים ומחלקות סטנדרטיות של Java שאינם בהכרח thread-safe:
Queue – ממשק של תור. המתודה ()remove מחזירה את האיבר הבא תוך הסרתו מן התור.
Stack – מימוש של מחסנית, עם מתודה ()lastElement המחזירה את האיבר העליון במחסנית מבלי להורידו.
Random –המתודה (nextInt(int i מחזירה מספר אקראי בתחום [0,i-1].

```
class CardTable {
  private Stack<Integer> _cards;
  CardTable() {
    _cards = new Stack<Integer>();
    _cards.add(5);
  }
  public synchronized int addCard(Integer card) throws WrongCardValueException {
    if (!legalCard(card))
      throw new WrongCardValueException(card);
    int ret = Math.abs(card - _cards.lastElement());
    _cards.add(card);
    return ret;
  }
  private static boolean legalCard(Integer card) {
    return (card > 0) && (card < 11);
  }
}
```

```
class Player implements Runnable {
  private CardTable _table;
  private Queue<Integer> _cards;
  private int _debt;
  Player(CardTable table, Queue<Integer> cards) {
    _table = table;   _cards = cards;   _debt = 0;
  }
  public void run() {
    Random ran = new Random();
    while (!_cards.isEmpty() && !Thread.currentThread().isInterrupted()) {
      Integer card = _cards.remove();
      try {
        _debt += _table.addCard(card);
        Thread.sleep(rand.nextInt(1000));  // Wait a random delay between 0 and 1 second
      } catch (InterruptedException e) {
        return;
      }
    }
  }
  public int getDebtPoints() { return _debt; }}
```

```
class Game {
  private static Integer getRandomCard() {
     Random rand = new Random();
     return rand.nextInt(10)+1;
  }
  private static Queue<Integer> createCards(int size) {
     Queue<Integer> ret = new LinkedList<Integer>();
     for (int i=0; i<size; i++)
       ret.add(getRandomCard());
     return ret;
  }
  public static void main(String[] args) throws InterruptedException, WrongCardValueException {
     CardTable table = new CardTable();
     Player player1 = new Player(table, createCards(32));
     Player player2 = new Player(table, createCards(32));
     Thread t1 = new Thread(player1);
     Thread t2 = new Thread(player2);
     t1.start();
     t2.start();
     t1.join();
     t2.join();
     if (player1.getDebtPoints() == player2.getDebtPoints())
       System.out.println("No winner for this game...");
     else if (player1.getDebtPoints() < player2.getDebtPoints())
       System.out.println("Player1 won this game");
     else
       System.out.println("Player2 won this game");
  }
}
```

א. הגדירו את התכונה הנשמרת (@inv) עבור לוח הקלפים (**CardTable**), וציינו האם הוא בטוח תחת חישוב מקבילי כלשהוא [4 נקודות].

ב. נסחו תנאי התחלה (@pre) וסיום (@post) למתודה **addCard** במחלקה **CardTable** [2 נקודות].

ג. כדי להפוך את המשחק למעניין יותר, ניתנת למתחרים האפשרות לצמצם את נקודות החובה, על ידי המתנה לקלף בראש הערימה התואם לקלף שבידי השחקן. שני קלפים יוגדרו כתואמים, אם ההפרש המוחלט ביניהם אינו עולה על 3.
אם עוברות 10 שניות ולא חל כל שינוי בלוח המשחק, מניח השחקן בהזדמנות הראשונה (= עם קבלת זמן cpu) את הקלף שברשותו בראש הערימה, גם אם הוא אינו תואם לקלף בראש הערימה.
עדכנו את הקוד כך שיתמוך באסטרטגיית משחק זו. [8 נקודות].

בגרסה אחרת של המשחק, אין כל אפשרות להניח קלף שאינו תואם לקלף בראש הערימה.
ד. הדגימו כיצד עלולים חוקי המשחק בגרסה החדשה להביא לחבק deadlock [2 נקודות]

ה. ממשו פתרון לחבק, על ידי הוספה של אובייקט אקטיבי חדש למערכת: מנהל המשחק (Dealer). במידה ומתרחש החבק, מנקה מנהל המשחק את ערימת הקלפים שעל השולחן, מייצר קלף חדש כלשהוא, ומניחו על השולחן. מנהל המשחק מכיר את שולחן הקלפים, אך לא את הקלפים שביד השחקנים [12 נקודות].

ו. הגדירו מחדש את התכונה הנשמרת של לוח המשחק, בעקבות האסטרטגיה שהוצגה בסעיף הקודם (ניתן לענות על סעיף זה, גם אם החלטתם לדלג על הסעיף הקודם) [2 נקודות]

## שאלה 2 (30 נקודות)

נתונה התוכנית הבאה:

```
class Image {
public:
  virtual void print(Printer *p) {
    cout << "Printing is not supported" << endl;
  }
};

class Printer {
public:
  virtual void print(Image *img)=0;
  virtual void print(BWImage *bwImg) {
    cout << "Black & White Image is not supported" << endl;
  }
  virtual void print(ColorImage *colorImg) {
    cout << "Color Image is not supported" << endl;
  }
};

class BWImage : public Image {
  char imageData[3];
public:
  BWImage() {}
  virtual void print(Printer *p){
    cout << "BWImage - printMe" << endl;
    p->print(this);
  }
};

class RGB {
  char data[3];
};

class ColorImage : public Image {
```

```
  RGB imageData[3];
public:
  ColorImage() {}
  virtual void print(Printer *p) {
    cout << "ColorImage - printMe" << endl;
    p->print(this);
  }
};

class ColorPrinter : public Printer {
public:
  virtual void print(Image *img) {
    cout << "ColorPrinter - print" << endl;
    img->print(this);
  }
  virtual void print(ColorImage *img) {
    // printing colored img data
    cout << "ColorPrinter printing ColorImage" << endl;
  }
  virtual void print(BWImage *img) {
    // printing Black & White img data
    std::cout << "ColorPrinter printing BWImage" << std::endl;
  }
};
```

```
void doWork() {
  ColorPrinter p;
  Image *colorImg = new ColorImage();
  p.print(colorImg);
}
void main() {
  doWork();
}
```

ידוע כי הפלט של הרצת התוכנית הינו:

```
ColorPrinter - print
ColorImage - printMe
ColorPrinter printing ColorImage
```

א. במחלקה image המתודה print מוגדרת כך: **virtual void print(Printer* p)**
מה יהיה פלט התוכנית אילו היא הייתה מוגדרת כך: **void print(Printer* p)**
[3 נקודות]

ב. מה יהיה פלט התוכנית אם בשורות האחרונות של doWork נוסיף:
[5 נקודות]

```
Image *bwImg = new BWImage();
p.print(bwImg);
```

ג. מה יהיה פלט התוכנית אם במקום שה main יקרא ל doWork הוא תבצע את שורות הקוד הבא:
[5 נקודות]

```
ColorPrinter p;
ColorImage colorImg;
Image *pColorImg = &colorImg;
p.print(pColorImg);
```

ד. אייר את המחסנית והערימה (stack and heap) של התוכנית לאחר שהתבצעו שלוש השורות הראשונות של doWork המקורי (ללא שינוי).
את המחסנית יש לייצג כטבלה עם טור יחיד ובכל שורה יופיע נתון יחיד עם גודלו וערכו. את הערימה יש לייצג כמלבן יחיד ובו תתי מלבנים נוספים לפי הצורך המייצגים כל אחד נתון/אובייקט. יש לכלול באיור את גודל האובייקט ומיקומו בערימה. ניתן להניח כי מצביע תופס ארבעה בתים. כמו כן ניתן לבחור כתובות כלשהן לערכים בערימה. [12 נקודות]

ה. נשים לב כי בתוכנית המקורית שבשאלה יש דליפת זיכרון.
[5 נקודות]

על מנת לתקן את התוכנית נוסף הקוד הבא למחלקה **ColorImage** :

```
~ColorImage(){
  std::cout << "ColorImage - destructor" << std::endl;
}
```

ובסוף המתודה doWork נוספה השורה:

```
delete colorImg;
```

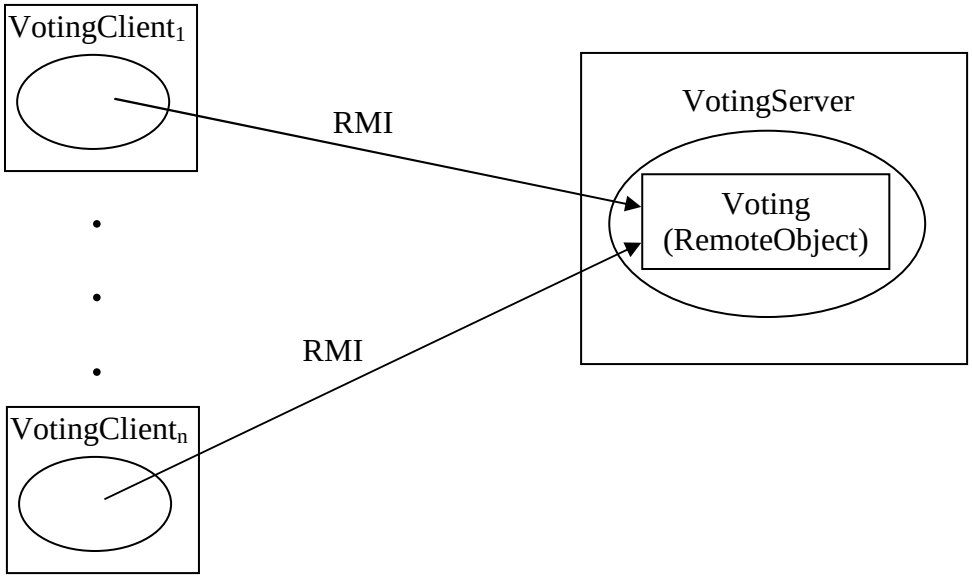
אך פלט ההרצה של התוכנית לא השתנה.

הסבירו ב-2 משפטים מדוע ומה צריך עוד לתקן בתוכנית על מנת לקבל את הפלט:

```
ColorPrinter - print
ColorImage - printMe
ColorPrinter printing ColorImage
ColorImage - destructor
```

## שאלה 3 (30 נקודות)

לקראת מערכת הבחירות הבאה, החליטה ועדת הכנסת לבדוק את אפשרות ההצבעה דרך מסופי מחשב. מנתחת מערכות - בוגרת המחלקה למדעי המחשב באוניברסיטת בן גוריון – הציעה את הארכיטקטורה הבאה:




הממשק Voting הינו ממשק Remote התומך בהצבעה למפלגה מסוימת, ובמתודה המחזירה את רשימת המפלגות.
השרת VotingServer מגדיר RemoteObject מסוג VotingImpl הממשש את הממשק Voting תוך שימוש במבנה נתונים מסוג Map הממפה את שם המפלגה למספר המצביעים עבורה.
מסוף ההצבעה VotingClient מתחבר ל RemoteObject כדי לבצע את הצבעות הבוחרים.

מנתחת המערכות החרוצה – בוגרת הקורס 'תכנות מערכות' – אף בנתה אבטיפוס, הממשש את המערכת באופן הבא:

```
public interface Voting extends java.rmi.Remote {
  Set<String> getParties() throws java.rmi.RemoteException;
  void vote(String party) throws java.rmi.RemoteException;
}
```

```
public class VotingImpl extends java.rmi.server.UnicastRemoteObject implements Voting {
  private final Map<String,Long> _mapParty2Votes;
  VotingImpl(String datafile) throws java.rmi.RemoteException {
    _mapParty2Votes = new TreeMap<String,Long>();
   // read line by line the list of parties running in this election from datafile
    try {
      BufferedReader in = new BufferedReader(new InputStreamReader(new FileInputStream(datafile),"UTF-8"));
      String party = null;
      while ((party = in.readLine()) != null)
         _mapParty2Votes.put(party,0L);
    } catch (Exception e) {
      throw new java.rmi.RemoteException(e.toString());
    }
  }
  public synchronized  Set<String> getParties() throws java.rmi.RemoteException {
    return _mapParty2Votes.keySet();   // return the set of keys in the map as a set of strings
  }
```

```
  public synchronized void vote(String party) throws java.rmi.RemoteException {
    Long votes = _mapParty2Votes.get(party);
    if (votes == null)
      throw new java.rmi.RemoteException("Party "+party+" is not running in this election");
    else {
       _mapParty2Votes.put(party,votes+1);
    }
  }
}
```

```
public class VotingServer {
  public static void main(String[] args) {
    Voting voting = null;
    try {
      voting = new VotingImpl("parties.txt");
      Naming.rebind( "//132.24.56.8:2002/Vote", voting);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```
public class VotingClient {
 public static void main(String[] args) {
  try {
    Voting voting = (Voting)Naming.lookup("//132.24.56.8:2002/Vote");
    Set<String> parties = voting.getParties();
    while (true) {
      System.out.println("Parties: " + parties);
      System.out.println("Please type the name of your favorite party, and press Enter");
      BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
      try {
         voting.vote(in.readLine());
      } catch (RemoteException e) {
          System.out.println(e.toString());
      }
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
 }
}
```

א. כדי להריץ את המערכת, איזו תוכנית נוספת יש להריץ תחילה? על איזה host ו port? [4 נקודות]

ב. כמה עותקים של רשימת המפלגות נוצרים כתוצאה מהרצת השורה הבאה (מתוך מתודת ה main של **VotingClient**)? נמקו תשובתכם. [4 נקודות]

```
Set<String> parties = voting.getParties();
```

בוגרת הקורס המצטיינת החליטה לכתוב בעצמה את מחלקות ה Skel וה Stub של **VotingImi**, במקום המחלקות הנוצרות אוטומטית על ידי rmic (הקומפיילר של RMI).
להלן המימוש שלה למחלקות אלו:

```
public class Voting_Skel {
  Voting_Skel(Voting voting) throws Exception {
    ServerProtocolFactory protocolMaker = new ServerProtocolFactory() {
      public AsyncServerProtocol create() {
        return new VotingProtocol(voting);
      }
    };
    final Charset charset = Charset.forName("UTF-8");
    TokenizerFactory tokenizerMaker = new TokenizerFactory() {
      public StringMessageTokenizer create() {
        return new FixedSeparatorMessageTokenizer("\n",charset);
      }
    };
    int port = 1984;
    int poolSize = 10;
    new Reactor(port, poolSize, protocolMaker, tokenizerMaker).start();
  }
}
```

```
public class VotingProtocol implements AsyncServerProtocol {
  private boolean _shouldClose, _connectionTerminated;
  private Voting _voting;
  VotingProtocol(Voting voting) {
    _shouldClose = false;
    _connectionTerminated = false;
    _voting = voting;
  }
  public String processMessage(String msg) {
    if (_connectionTerminated)
      return null;
    if (isEnd(msg)) {
      _shouldClose = true;
      return "CLOSED";
    }
```

```
    if (msg.startsWith("VOTE ")) {
      try {
        _voting.vote(msg.substring(5).trim());
        return "SUCCESS";
      } catch (RemoteException e) {
        return "FAIL";
      }
      } else if (msg.startsWith("GET_PARTIES")) {
        try {
          Set<String> parties = _voting.getParties();
        StringBuilder sb = new StringBuilder();
        for (String party : parties) {
          sb.append(party);
          sb.append("#");
        }
        return sb.toString();
      } catch (RemoteException e) {
        return "FAIL";
      }
    }
    return "FAIL";
  }

  public boolean isEnd(String msg) {  return msg.equals("BYE");  }
  public boolean shouldClose() {   return _shouldClose;   }
  public void connectionTerminated() {   _connectionTerminated = true;   }
}
```

```
public class Voting_Stub implements Voting {
  String  _skelHost;
  int _skelPort;

  Voting_Stub(String skelHost,int skelPort) throws RemoteException {
    _skelHost = skelHost; _skelPort = skelPort;
  }

  public Set<String> getParties() throws RemoteException {
    //@TODO – סעיף ג
   }

  public void vote(String party) throws RemoteException {
    try {
      Socket socket = new Socket(_skelHost,_skelPort);
      String msg = "VOTE " + party + "\n";
      socket.getOutputStream().write(msg.getBytes("UTF-8"));
```

```
      BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream(),"UTF-8"));
      String ans = in.readLine();
      msg = "BYE\n";
      socket.getOutputStream().write(msg.getBytes("UTF-8"));
      socket.close();
    } catch (Exception e) {
      throw new RemoteException(e.toString());
    }
    if  (!ans.equals("SUCCESS"))
      throw new RemoteException("Vote failed");
  }
}
```

ג. השלימו את הקוד החסר במתודה **getParties()** במחלקה **Voting_Stub** (האחרונה בקוד למעלה). [6 נקודות]

ד. ענו בקצרה על השאלות הבאות (על סמך הכרותכם את ה reactor שנלמד בכיתה): [6 נקודות]

- כמה מסופי הצבעה (**VotingClient**) יכולים להיות מחוברים ל **VotingImpl**?
- כמה בקשות הצבעה יכולות להיות מטופלות במקביל ב **VotingProtocol**?
- כמה בקשות הצבעה יכולות להתבצע במקביל במתודה **vote** במחלקה **VotingImpl**?

במערכת שתוארה עד כה, בוחר המצביע במפלגתו, כאשר האחריות לבדיקת הרשאתו להצביע נשארת בידי ועדת הקלפי. כעת נדרש להרחיב את המערכת כך שהיא תאחסן את נתוני הבוחרים, ותבדוק את הרשאתם להצביע.

ה. הממשק Voting עודכן: המתודה vote מקבלת כעת פרמטר המציין את מספר הזהות של המשתמש, כך שהוא יוכל להצביע רק אם הוא מופיע בבסיס הנתונים, ובתנאי שהוא לא הצביע עד כה:

```
public interface Voting extends java.rmi.Remote {
  Set<String> getParties() throws java.rmi.RemoteException;
  void vote(String party, long userID) throws java.rmi.RemoteException;
}
```

עדכנו את המחלקות **VotingImpl**, **Voting_Stub**, **Voting_Skel**, **VotingProtocol** כך שיתמכו בממשק החדש [10 נקודות]

## שאלה 4 (10 נקודות)

המערכת שנבנתה בשאלה הקודמת הוצגה בפני ועדת הבחירות הממלכתית. הוועדה החליטה לאשר את המערכת בתנאי שנתוני ההצבעה ישמרו במערכת לניהול בסיסי נתונים SQL.

א. הגדירו מודל נתונים שיכיל את נתוני הבוחרים (מספר זהות, מספר קלפי, האם הצביע?), קלפיות (מספר קלפי, שם ישוב, שעת פתיחה), ומפלגות (שם המפלגה, מספר הצבעות).
הגדרת המודל תתבסס על טבלאות ומפתחות (ראשי,זר). [5 נקודות]

ב. כתבו שאילתה המחזירה את מספרי הזהות של הבוחרים שלא הצביעו עדיין בקלפי מספר 17. [5 נקודות]