גיליון תשובות

מספר נבחן:_____

Q1	Q2
Q3	Q4
Q5	TOTAL

שאלה 1 (30 נקודות)

סעיף א (6 נקודות)

האיווריאנטה נשמרת והמחלקה בטוחה.
במערכת המתוארת בשאלה קיים ת'רד left אחד בלבד, ות'רד right אחד בלבד, בנוסף נתון כי רק ת'רד left מבצע moveLeft() ורק ת'רד right מבצע moveRight(). העולה מכך הוא שלא יתכן מצב בו ת'רד left קורא x=2_ המקיים את תנאי ה if, ת'רד left אחר מבצע moveLeft() ומשנה את x_ ל 1, ואז הת'רד הראשון מבצע x_-- לערך הלא חוקי 0. וכן להפך, שלא יתכן מצב בו ת'רד right קורא x=9_ המקיים את תנאי ה if, ת'רד right אחר מבצע moveRight() ומשנה את x_ ל 10, ואז הת'רד הראשון מבצע x_++ לערך הלא חוקי 11.

סעיף ב (6 נקודות)

המחלקה אינה בטוחה. לדוגמא:
נתון: x_ = 5
קלט: (left,right)

פלטים אפשריים בחישוב המקבילי:
(4,5) [ת'רד left מבצע moveLeft() ואחריו ת'רד right מבצע moveRight()]
(6,5) [ת'רד right מבצע moveRight() ואחריו ת'רד left מבצע moveLeft()]
(6,4) [ת'רד left קורא את ערכו של x_ (5), ת'רד right קורא את ערכו של x_ (5), ת'רד right מעדכן את ערכו של x_ ל 6, ת'רד left מעדכן את ערכו של x_ ל 4]
(4,6) [ת'רד left קורא את ערכו של x_ (5), ת'רד right קורא את ערכו של x_ (5), ת'רד left מעדכן את ערכו של x_ ל 4, ת'רד right מעדכן את ערכו של x_ ל 6]

הפלטים האפשריים בחישוב הסדרתי: (4,5) עבור הקלט (left,right)
(6,5) עבור הקלט (right, left)

לפלטים (6,4) ו (4,6) אין מקבילה בחישוב הסדרתי בכל סדר של הקלט.

סעיף ג (6 נקודות)

```
MoveLeft:

@PRE:       None beyond the @inv (MIN_X <= x_ <= MAX_X)
@POST:      x_ == MIN_X || x_ == @pre(x_) - 1
```

```
MoveRight:

@PRE:       None beyond the @inv (MIN_X <= x_ <= MAX_X)
@POST:      x_ == MAX_X || x_ == @pre(x_) + 1
```

תנאי ההתחלה מתקיימים באופן ריק (או לדרישות האינווריאנטה, כפי שהראנו בסעיף א).
תנאי הסיום לא מתקיימים תמיד כפי שהראנו בסעיף ב.

סעיף ד (12 נקודות)

נשתמש בסנכרון כדי להבטיח התאמת הפלט המקבילי לסדרתי בסדר כלשהוא, ולשם קיום תנאי הסיום של המתודות moveRight(), moveLeft(), moveDown(), moveUp().

נשתמש במנגנון wait/notify כדי להבטיח קיום תנאי ההתחלה של המתודות
moveLeft()moveRight(), moveUp()

```
class Firemen
{
   public static final int MIN_X = 1;
   public static final int MAX_X = 10;
   public static final int MIN_Y = 1;
   public static final int MAX_Y = 3;
   private int x_,y_;

   // @INV: MIN_X <= x_ <= MAX_X && MIN_Y <= y_ <= MAX_Y &&
   //       (y_ = 1 || 4 > x_ > 6)
   Firemen() { x_ = 5; y_ = 1; }

   public synchronized void repaint()
   {
      //code that paints the firemen and the stretcher at position x_
   }

   public synchronized void moveLeft()
   {
      try {
            while (x_ == 7 && y_ > MIN_Y)
               wait();
            if (x_ > MIN_X) {
               x_--;
               notifyAll();
             }
      } catch (InterruptedException e) {
      }
   }

   public synchronized void moveRight()
   {
      try {
            while (x_ == 3 && y_ > MIN_Y)
               wait();
            if (x_ < MAX_X) {
               x_++;
               notifyAll();
             }
      } catch (InterruptedException e) {
      }
   }
```

```
   public synchronized void moveUp()
   {
      try {
            while ((x_ > 3 && x_ < 7) && y_ > MIN_Y)
               wait();
            if (y_ < MAX_Y)
               y_++;
      } catch (InterruptedException e) {
      }
   }

   public synchronized void moveDown()
   {
      if (y_ > MIN_Y) {
         y_--;
         notifyAll();
      }
   }

}
```

שאלה 2 (30 נקודות)

סעיף א (5 נקודות)

The invariant is:
- The value of all the elements in the vector is between 1 and maxval_
- The vector has a size between 0 and maxsize_
- The vector does not contain the same value more than once

```
bool intset::inv() {
        if (elements_.size() > maxsize_) return false;
        if (hasRepetitions(elements_) return false;
        for (vector<int>::const_iterator it=elements_.begin(); it!=elements_.end(); ++it) {
                if (*it > maxval_) || (*it < 1) return false;
        }
        return true;
}
```

סעיף ב (5 נקודות)

The @precond of the constructor is:
- m >= 0
- n >= 1
- m <= n (to satisfy the constraint that any instance of the class may eventually become full)

The code of the invariant is:

```
intset::intset(int m, int n) : maxsize_(m), maxval_(n) {
      if (m < 0 || n < 1 || m > n) {
            throw exception("Invalid intset parameters");
      }
}
```

NOTE: It is ok to throw exceptions in constructors. When an exception is thrown from a constructor, the object is not constructed – it never becomes an object, since the constructor never returns.

On the other hand, it is NEVER ok to throw an exception in a destructor.

סעיף ג (2 נקודות)

The destructor is empty – there are no resources acquired in the lifetime of this object.
The STL vector (and all STL containers) behaves as a value – not as a reference. It is copied as a value, allocated as a value.

```
intset::~intset() {}
```

סעיף ד (6 נקודות)

```
bool intset::member(int t) {
  // @pre: no precondition
  // @post: returned value is true if and only if t belongs to elements_
  if (t < 1 || t > maxval_) return false;
  for (vector<int>::const_iterator it=elements_.begin();
       it!=elements_.end(); ++it) {
      if (*it == t) return true;
  }
  return false;
}
```

NOTE1:
- The @pre and @post do not include the invariant condition – which must always hold before @pre and after @post. @pre and @post are constraints specific to the method.

- It is possible to define the constraint that (1 <= t <= maxval_) as a precondition. This is a design decision (not a very good one – but one that could be considered in certain conditions). If this is the case, then the code must be changed so that the precondition failure throws an exception.

```
void intset::insert(int t) {
      // @pre: 1 <= t <= maxval_ and elements_.size() < maxsize_
      // @post: member(t) and
      // if @pre(member(t)) then (elements_size()=@pre(elements_.size()))
      // if @pre(!member(t))then (elements_.size()=pre(elements_.size())+1
      if (t < 1 || t > maxval_) throw exception("value out of range");
      if (elements_.size() >= maxsize_) throw exception("set is full");
      if (!member(t)) elements_.push_back(t);
}
```

NOTE2:

- It is here required to check the value of t as part of @precond
- It is possible to consider that inserting a value which is already in the set is a precond violation. This is a design decision (not a good one – but it could be required in certain conditions). If this is the case, then the code must be changed so that the precondition failure throws an exception.

סעיף ה (3 נקודות)

```
class intset {
private:
      const int maxsize_; // How many elements the set can store
      const int maxval_;  // All elements values must be in [1,maxval_]
      vector<int> elements_;
public:
      // MUST use the initialization form because the fields are constant
      intset(int m, int n) : maxsize_(m), maxval_(n) {
            // same as in 2.2
      }
      ~intset() {};
      bool member(int t) const;     // is t a member? Does not change the
                                    // state of the object
      void insert(int t);           // add t to set
};
```

Note:
- There is no reason to mark int params passed by value as const.
- There is no reason to mark returned value bool by value as const.
- The elements_ member is not constant because we must apply methods such as push_back

סעיף ו (9 נקודות)

```
{
      intset s0(3, 5);
      intset* s1 = new intset(2, 5);
      intset s2(10, 5);             // Exception (m > n)
      intset s3;                    // Compilation error: no default ctor
      s0.insert(1);
      s0.insert(1);                 // Depends on your decision in 2.4
      s0.insert(2);
      s0.insert(3);
      s0.insert(5);                 // Exception "set is full"
      s0.insert(6);                 // Exception "out of range"
                                    // depends on order of verifications in 2.4
      *s1 = s0;                     // THIS IS OK (copy of value)
      {
         intset s4(1, 5);
         s4.insert(3);
         s0 = s4;                   // THIS IS OK (copy of value)
      }
      s0.member(1);                 // THIS IS OK (s0 has a valid value)
      // MISSING delete s1;
}
```

שאלה 3 (12 נקודות)

סעיף א (4 נקודות)

התוכנית client תדפיס את ערכו המקורי של element – 2.
Element הוא אובייקט רגיל ועל העברתו תתבצע by value – כלומר ה serializtion הוא על האובייקט, כך שכל השינויים עליו מבוצעים בזיכרון של התהליך אליו הוא נשלח (במקרה הזה ה במקרה הזה ה server).

סעיף ב (8 נקודות)

נשנה את הגדרת הממשק Increasable והמחלקה IncNumber כך שיגדירו remote object. על ידי כך element ישלח by reference – כלומר ה serialization הוא על ה stub , וכל השינויים יבוצעו בזיכרון של התהליך בו הוא הוגדר (במקרה זה ה client).

```
interface Increasable
  extends java.rmi.Remote
{
  public void increase()
    throws java.rmi.RemoteException;
}

class IncNumber
  extends java.rmi.server.UnicastRemoteObject
  implements Increasable
{
   protected long num_;
   protected long offset_;

   public IncNumber(long n, long o)
     throws java.rmi.RemoteException
   {
      num_ = n;
      offset_ = o;
   }

   public synchronized void increase()
     throws java.rmi.RemoteException
     {
        num_ += offset_;
     }

   public long getNum()   { return num_;  }
}
```

שאלה 4 (18 נקודות)

סעיף א (3 נקודות)

אין וודאות שכל הבתים בהודעת המוחזרת response אכן ייכתבו ל _channel , שהרי פעולת write היא non

Blocking - היא כותבת את מספר הבתים שניתן לכתוב בשלב זה.

סעיף ב (5 נקודות)

נוסיף לולאה המבצעת write עד אשר נכתבו כל הבתים של response ל _channel:

```
public void executeTask() throws TaskFailedException {
   String response = "Got your message!;";
   int iBytesToWrite = response.getBytes().length;
   int iWrittenBytes = 0;
   try {
      while (iWrittenBytes < iBytesToWrite)
      iWrittenBytes += _channel.write(ByteBuffer.wrap(
                              response.substring(iWrittenBytes).getBytes()));
   }
   catch (IOException io) {
     throw new TaskFailedException(
          "I/O exception while processing the message: " + _message, io);
   }
}
```

סעיף ג (10 נקודות)

הודעות המתקבלות ב _sChannel נקראות ב ConnectionHandler בחלקים: בכל פעם שחל ארוע OP_READ ב _sChannel הבתים הניתנים לקריאה בשלב זה נקראים על ידי המתודה read() המאחסנת אותם בשדה _incomingData (כאשר _incomingData מכיל הודעה שלמה, היא מועברת ל ThreadPool לביצוע כ MessageProcessorTask).

בדיוק באותו אופן נטפל בהודעות המיועדות לכתיבה ב _sChannel :

- הוספת פעולת כתיבה ב ConnectionHandler
  - נגדיר ב ConnectionHandler שדה נוסף _outgoingData המחזיק את הבתים המיועדים לכתיבה ב _sChannel. בכל פעם שנדרשת כתיבת תשובה להודעה שנקבלה (במתודה executeTask של MessageProcessorTask), היא מוספת ל _outgoingData.
  - נגדיר מתודה חדשה write() הכותבת ל _sChannel בתים מ _outgoingData.
- רישום הערוץ _sChannel ב _selector לאירועי כתיבה, והוספת תגובה מתאימה.
  - כאשר נדרשת כתיבת התשובה לבקשה במתודת executeTask(), יש לדאוג לכך שה selector יזהה אירוע OP_WRITE ב _sChannel (המציין את האפשרות לכתוב ל output stream של הערוץ) על ידי ביצוע register עם OP_READ | OP_WRITE.
  - אירוע של אפשרות לכתיבה, יזוהה ב Reactor ויגרור הפעלה של המתודה write() ב ConnectionHandler (בדומה לזיהוי אפשרות קריאה הגוררת הפעלה של מתודת read() ב ConnectionHandler).
  - כאשר לא נשארו בתים לכתיבה ב _outgoingData, יש להסיר את אירוע הכתיבה עבור _sChannel ב _selector, על ידי ביצוע register מתאים עם OP_READ בלבד.

להלן הקוד המעודכן, ההוספות צבועות באדום:

```
Class Reactor extends Thread
```

```
{
...
public void run() {
   try {
      _pool = new ThreadPool(_poolSize);
      _pool.startPool();

      ServerSocketChannel ssChannel = ServerSocketChannel.open();
      ssChannel.configureBlocking(false);
      ssChannel.socket().bind(new InetSocketAddress(_port));
      Selector selector = Selector.open();
      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()) {
               ConnectionHandler connectionHandler =
                                    (ConnectionHandler)selKey.attachment();
               connectionHandler.read();
            }

            if (selKey.isValid() && selKey.isWritable()) {
               ConnectionHandler connectionHandler =
                                   (ConnectionHandler)selKey.attachment();
               connectionHandler.write();
            }
         }
      }
   } catch (Exception e) {
      e.printStackTrace(System.err);
      stopReactor();
   }
   stopReactor();
...
}



public class ConnectionHandler {
    public static final int BUFFER_SIZE = 256;
    public static final char MESSAGE_END = ';';

    protected SocketChannel _sChannel;
    protected String _incomingData;
    protected StringBuffer _outgoingData;
    protected ThreadPool _pool;
    protected Selector _selector;

    public ConnectionHandler
```

```
        (SocketChannel sChannel, ThreadPool pool,Selector selector)
    {
        _sChannel = sChannel;
        _pool = pool;
        _selector = selector;
        _incomingData = "";
        _outgoingData = new StringBuffer();
    }

    public void read() throws IOException {
       SocketAddress address = _sChannel.socket().getRemoteSocketAddress();
       System.out.println("Reading from " + address);
       ByteBuffer buf = ByteBuffer.allocate(BUFFER_SIZE);
       while (true) {
          buf.clear();
          int numBytesRead = _sChannel.read(buf);
          if (numBytesRead == -1) {
             _sChannel.close();
             break;
          }
          if (numBytesRead > 0) {
             buf.flip();
             String str = new String(buf.array(), 0, numBytesRead);
             _incomingData = _incomingData + str;
          }
          if (numBytesRead < BUFFER_SIZE)
             break;
       }

       while (true) {
           int pos = _incomingData.indexOf(MESSAGE_END);
           if (pos==-1)
               break;
           String message = _incomingData.substring(0, pos);
           _incomingData = (pos==_incomingData.length()-1) ?
                           "" : _incomingData.substring(pos+1);

          _pool.addTask(new MessageProcessorTask(message,
                                        _sChannel,_outgoingData,this));
       }
    }

    //Writes replies to the client
    public void write() throws TaskFailedException {
      if (_outgoingData.length() > 0) {
      int numBytesWrite = 0;
      try {
         ByteBuffer buffer =
                    ByteBuffer.wrap(_outgoingData.toString().getBytes());
         numBytesWrite = _sChannel.write(buffer);
      }
      catch (IOException io) {
        throw new TaskFailedException("I/O exception while writing: " +
                                                  _outgoingData, io);
      }
      if (numBytesWrite > 0)
         _outgoingData.delete(0,numBytesWrite);
      if (_outgoingData.length() == 0)
      try {
         readRegistration();
      } catch (java.nio.channels.ClosedChannelException e) {
```

```
            throw new TaskFailedException(e.toString());
         }
      }

   // Registers to the selector on read and write events.
   // Selector's waking up is done in order to synchronize the new
   // registration with other threads which wait on select()command
   public void writeRegistration() throws
                                 java.nio.channels.ClosedChannelException
   {
      _sChannel.register(_selector,
                         SelectionKey.OP_READ | SelectionKey.OP_WRITE,this);
      _selector.wakeup();
   }

   // Registers to the selector on read events.
   // Selector's waking up is done in order to synchronize the new
   // registration with other threads which wait on select() command
   public void readRegistration() throws
                                   java.nio.channels.ClosedChannelException
   {
      _sChannel.register(_selector,SelectionKey.OP_READ,this);
      _selector.wakeup();
   }
}


class MessageProcessorTask implements Task {
   protected String _message;
   protected SocketChannel _channel;
   protected StringBuffer _outgoingData;
   protected ConnectionHandler _connectionHandler;
   public MessageProcessorTask(String message, SocketChannel channel,
      StringBuffer outgoingData,ConnectionHandler connectionHandler) {
      _message = message;
      _channel = channel;
      _outgoingData = outgoingData;
      _connectionHandler = connectionHandler;
   }


   public void executeTask() throws TaskFailedException {
      String response = "Got your message!;";
      synchronized(_outgoingData) {
         _outgoingData.append(response);
      }
      try {
         _connectionHandler.writeRegistration();
      } catch (java.nio.channels.ClosedChannelException e) {
         throw new TaskFailedException(e.toString());
      }
   }
}
```

המהדרין מן המהדרין, מעדכנים גם את ה ConnectionAcceptor - העברת ה Selector כפרמטר לבונה של ConnectionHandler.

שאלה 5 (10 נקודות)

סעיף א (7 נקודות)

Words

Primary Key	Str
	Freq

Analyses

Primary Key	ID
	POS
	Gender
	Number

WordsAnalyses

Primary Key	WordStr
	AnalysisId

WordStr מפתח זר לשדה Str ב Words, ו AnalysisId מפתח זר לשדה ID ב Analyses.

סעיף ב (3 נקודות)

```
SELECT      Words.Str
FROM        Words
WHERE       Words.Freq > 10000
ORDER BY    Words.Freq desc
```