גיליון תשובות

מספר נבחן:_____

Q1	Q2
Q3	Q4
Q5	Q6
TOTAL	

שאלה 1 (30 נקודות)

סעיף א (10 נקודות)

```
Room 0 is serviced
Room 1 is serviced
Room 1 is serviced
או
Room 0 is serviced
Room 1 is serviced

או כלום
```

סעיף ב (10 נקודות)

```
public void synchronized requestService() {
  while (isServiceNeeded_()) {
     try {
        wait();
      } catch (java.lang.InterruptedException e) {
      }
   }
   serviceNeeded_ = true;
}

public void synchronized serviceDone() {
   serviceNeeded_ = false;
   notifyAll();
}
```

סעיף ג (10 נקודות)

במחלקה Room יש לתמוך במצב בו לא ניתן לבקש בקשות נוספות (עבור פרק הזמן שאחר ביצוע ה interruption). בנוסף יש לזהות מתי הסתיים ביצוע הבקשה האחרונה עבור חדר זה – כולל בקשות שבהמתנה. לשם כך יש להגדיר counter הגדל לפני כל המתנה להעלאת בקשה ואשר יורד בסיום ביצוע בקשה.

```
class Room {
```

```
   private String Id_;
   private int reqCount_;
   private boolean reqMode_;
   private boolean serviceNeeded_;

   public Room(String name) {
      Id_ = name;
      serviceNeeded_ = false;
      reqMode_ = true;
      reqCount_ = 0;
   }
   public String synchronized getName(){return Id_; }
   public boolean synchronized existReq () { return reqCount_ > 0; }
   public boolean synchronized isServiceNeeded() {
       return serviceNeeded_;
   }

   public void synchronized requestService() {
      if (reqMode_)
         reqCount_++;
         while (isServiceNeeded()) {
            try {
               wait();
            } catch (java.lang.InterruptedException e) {
            }
         }
         serviceNeeded_ = true;
      }
   }

   public void synchronized serviceDone() {
     serviceNeeded_ = false;
     reqCount_--;
     notifyAll();
   }

   public void synchronized disableReq() { reqMode_ = false;  }
}
```

במחלקה RoomService, עבור אירוע של interrupt נעביר את כל החדרים למצב בו לא ניתן להעלות בקשות נוספות, ונמשיך את תהליך מילוי הבקשות עד סיומן אצל כל החדרים.

```
class RoomService extends Thread {
    private Vector rooms_;
    RoomService(Vector rooms)  { rooms_ = rooms; }
    public void run()  {
        boolean existReq = false;
        boolean reqMode = true;

        while (reqMode || existReq) {
           try {
              existReq = false;
              sleep(1000);
              for (int j=0;  j<rooms_.size();  j++)   {
                 Room room = (Room)rooms_.get(j);
                 if (room.isServiceNeeded()) {
                    service(room);
                    existReq = existReq || room.existReq();
                 }
              }
```

```
            } catch (java.lang.InterruptedException ie) {
               reqMode = false;
               for (int j=0;  j<rooms_.size();  j++) {
                  Room room = (Room)rooms_.get(j);
                  room.disableReq();
                  existReq = existReq || room.existReq();
               }
            }
         }
    }

    private void service(Room room)    {
      // service is performed here…
      room.serviceDone();
      System.out.println("Room " + room.getName() + " is serviced");
    }

    public static void main(String args[]) {
        Vector rooms = new Vector();
        rooms.add(new Room("room 0"));
        rooms.add(new Room("room 1"));
        for (int i = 0; i < rooms.size(); i++) {
            ((Room)rooms.get(i)).requestService();
        }
        RoomService S1 = new RoomService(rooms);
        S1.start();
        ((Room)rooms.get(1)).requestService();
        try {
            sleep(2000);
        } catch (java.lang.InterruptedException ie) {
        }
        S1.interrupt();
   }
}
```

שאלה 2 (15 נקודות)

סעיף א (5 נקודות)

כפי שנלמד בכיתה, עבור מחלקה יורשת, מתבצע קודם תהליך ההריסה עבור הבן ואח"כ מופעל אוטומטית תהליך ההריסה עבור האב, ובמקרה שלנו יודפס:

```
B: destructor
A: destructor
```

סעיף ב (10 נקודות)

כפי שלמדנו בכיתה, עבור מחלקה יורשת, מתבצע קודם האיתחול של מחלקת האב ורק אחר כך של מחלקת הבת. במידה ולא מצוין בונה מסוים עבור מחלקת האב מופעלת ברירת המחדל.
במקרה שלנו בבונה של B אין אתחול מסוים לבונה של A ולכן מופעל בונה ברירת המחדל של A המייצר מקום על ה heap עבור p ומשים למקום אליו הוא מצביע את הערך 0. לאחר מכן בביצוע הבונה של B מוקצה זיכרון חדש והכתובת מושמת ל p_ כך שהזיכרון שהוקצה קודם הולך לאיבוד.

כדי לפתור בעיה זו, יש לתת את הדעת על כך שהקצאת הזיכרון בבונה של B מיותרת, היא באחריות המחלקה A שמגדירה את p_, ועל כן B בסה"כ ישים את הערך הנדרש במקום אותו הקצתה המחלקה A :

```
class B : public A {
public:
      B(int i) { *p_ = i; }
      virtual ~B() {  std::cout << "B: destructor"; }
};
```

שאלה 3 (15 נקודות)

סעיף א (10 נקודות)

1.א

```
Class Enumerator {
   private int i_;
   Enumerator(int i) { i_ = i; }
   int next { return ++i_; }
}

boolean F1(Enumerator e) {
      return e.next() < 100;
}
```

2.א

הפונקציה F2 מקדמת את הפרמטר האקטואלי שלה i.

i הוא מצביע, וקידומו באחד יגרום לכך שהוא יצביע לתא הבא בזיכרון - גישה ישירה לזיכרון אינה אפשרית ב Java.

סעיף ב (5 נקודות)

```
class I  {
public:
   virtual boolean F() = 0;
};

class A {
public:
   A(int i) { i_ = i; }
protected:
   int  i_;
};

class B : public A, public I {
public:
   B(int i) : A(i) {}
   virtual boolean F()   { return i_ < 100; }
};
```

שאלה 4 (15 נקודות)

סעיף א (5 נקודות)

2 פעולות הלוך-חזור: אחת עבור getAddress() ואחת עבור getPhoneNumber()

סעיף ב (5 נקודות)

```
public interface RemoteAddressInterface  extends java.rmi.Remote
{
  public String getPhoneNumber() throws java.rmi.RemoteException;
  public String getAddress() throws java.rmi.RemoteException;
  public String getAddressAndPhoneNumber()
       throws java.rmi.RemoteException;
}

Public String getData(String familyName)
          throws UnknownFamilyException, java.rmi.RemoteException
{
   RemoteAddressInterface  family;
   family = mapFamilyname2RemoteAddress.get(familyName);
   if (family == null)
      throw new UnknownFamilyException(familyName);

   return "Family: " + familyName +
           ", Details: " + family.getAddressAndPhoneNumber();
}
```

ניתן ללכת בדרך אלגנטית יותר ולהגדיר פונקציה המחזירה ווקטור של נתונים:

```
public interface RemoteAddressInterface  extends java.rmi.Remote
{
  public Vector getData () throws java.rmi.RemoteException;
}

Public String getData(String familyName)
          throws UnknownFamilyException, java.rmi.RemoteException
{
   RemoteAddressInterface  family;
   family = mapFamilyname2RemoteAddress.get(familyName);
   if (family == null)
      throw new UnknownFamilyException(familyName);

   String str = new String("Family: " + familyName + ", ");
   Vector v = family.getData();
   for (int i=0; i<v.size(); i++)
      str += (String)v.get(i);

   return str;
}
```

תשובות בהם שונה הממשק מ remote ל serializable קיבלו ניקוד חלקי, שכן במקרה זה המידע המתקבל אינו מעודכן – הממשק מחזיר את פרטי המשפחה כפי שהיו בזמן שהיא נרשמה לשרות (משל למה הדבר דומה: דמיינו כי החנויות בתרגיל 3 מגישות עותק שלהם לשרת, כך שברור מחירו של מוצר מתבסס על העותק ולא על המידע בחנות עצמה בזמן אמת, תוך התעלמות מהעלאת מחירים, או למצער, ממבצעים מיוחדים) אלא אם כן נוספה לממשק שיטה המאפשרת למשפחה לעדכן נתוניה אצל שרת הכתובות.

סעיף ג (5 נקודות)

(i)

```
public interface RemoteAddressInterface  extends java.rmi.Remote
{
  public String getPhoneNumber() throws java.rmi.RemoteException;
  public String getAddress() throws java.rmi.RemoteException;
  public String getEmail()  throws java.rmi.RemoteException;
}
```

(ii)

```
public interface RemoteAddressInterface  extends java.rmi.Remote
{
  public String getPhoneNumber() throws java.rmi.RemoteException;
  public String getAddress() throws java.rmi.RemoteException;
  public String getEmail()  throws java.rmi.RemoteException;
  public String getAddressPhoneNumberAndEmail()
      throws java.rmi.RemoteException;
}
```

הגישה הראשונה מודולרית יותר אך לא יעילה, והיפוכה בשיטה השנייה. העיצוב האלגנטי של החזרת ווקטור, מודולרי ויעיל.

שאלה 5 (15 נקודות)

סעיף א (5 נקודות)

על ידי ה checksum המצורף לכל יחידה

סעיף ב (5 נקודות)

78 יחידות יישלחו בשנית (10-87).

סעיף ג (5 נקודות)

<u>אפשרות א</u>: אלגוריתם selective-repeat אשר עיקר ענינו בשמירת הודעות, שהגיעו כתיקונן אך לא בסדר הנכון, בשכבת ה transport, תוך שחרורן לשכבת האפליקציה עם הגעתן התקינה של כל ההודעות הקודמות להן.

<u>אפשרות ב</u>: הפעלת אלגוריתם go-back-N עבור N=1 (שזה למעשה אלגוריתם stop & wait)

שאלה 6 (10 נקודות)

סעיף א (5 נקודות)

```
SELECT  Cows.NickName
FROM    Cows,Production
WHERE   Cows.ID = Production.ID AND
        Production.Year = 2003 AND
        Production.DayInYear = 3 AND
        Production. MilkQuantity = 70
UNION
SELECT  Cows.NickName
FROM    Cows,Weight
WHERE   Cows.ID = Weight.ID AND
        Weight.Year = 2003 AND
        Weight.DayInYear = 3 AND
        Weight. Weight = 134
```

תשובות שהתבססו על פעולת or בין שני חלקי השאילתא, תחת הגדרת union של שתי שאילתות, קיבלו את מלוא הנקודות על אף היותן שגויות.

סעיף ב (5 נקודות)

```
SELECT Cows.NickName
FROM   Cows,Production
WHERE  Cows.ID = Production.ID AND
       Production.Year = 2003 AND
       Production.DayInYear = 3 AND
       ((Cows.type = 1 AND Production.Data = 70) OR
        (Cows.type = 2 AND Production.Data = 134))
```