גיליון תשובות

מספר נבחן:_____

שאלה 1 (30 נקודות)

סעיף א: תוכנית מקבילית שאינה בטוחה, היא תוכנית בה מצב האובייקטים אינו זהה למצבם בהרצה סדרית כלשהי. להלן דוגמא לריצה שכזו:
נניח שהכדור נמצא ב (1,2). וכי התרד האופקי מנסה להזיזו שמאלה ואילו תרד הרוח מנסים להזיזו ימינה.
כלומר קיימות שתי פעולות: left, right
בהרצה סדרתית, בין אם נבצע קודם left ואח"כ right, ובין אם נבצע אותם בסדר הפוך: קודם right ואחר כך left, הכדור יחזור למקומו (1,2).
בהרצה מקבילית, לעומת זאת, מאחר ואין רצף סנכרון בין קריאת מקום הכדור וכתיבת המיקום המעודכן, יתכנו גם התוצאות (0,2) ו (2,2).
לדוגמא: נניח תזמון בו התרד האופקי קורא את ערכו האופקי של הכדור (כלומר קורא את הערך 1). נשים לב כי כעת העתק של ערך זה נמצא על רגיסטר וכל שינוי שיבוצע לערך זה יתבצע מקומית ולא יכתב באופן מיידי לזיכרון המשותף. כעת תרד הרוח מבצע את אותה הפעולה, כלומר קורא את הערך 1 ומעתיקו לרגיסטר כלשהו. הת'רד האופקי מוריד 1 וכותב 0, ואח"כ ת'רד הרוח מוסיף אחד וכותב 2.

על מנת לפתור בעיה זו, ניתן להשתמש בפתרון שהוצג במועד א' - סינכרון על מפתח אופקי או אנכי בהתאם לפעולה אשר אמורה להתבצע.

השינויים בקוד:

```
class SimpleBlockedBall extends CheckedBlockedBall {
...
public void up() throws Exception {
synchronized(_ballY){
if (!checkPreCondUp())
        throw new Exception("The pre-condition for the up() method does not hold!");
_ballY++;
}
}

public void down() throws Exception {
synchronized(_ballY){
if (!checkPreCondDown())
        throw new Exception("The pre-condition for the down() method does not hold!");
_ballY--;
}
}

public void right() throws Exception {
synchronized(_ballX){
if (!checkPreCondRight())
```

```
        throw new Exception("The pre-condition for the right() method does not hold!");
_ballX++;
}
}

public void left() throws Exception {
synchronized(_ballX){
if (!checkPreCondLeft())
        throw new Exception("The pre-condition for the left() method does not hold!");
_ballX--;
}
}


...
Integer _ballX, _ballY;
}
```

סעיף ב:

```
class SimpleBlockedBall extends CheckedBlockedBall {


...

public void up() throws Exception {
        synchronized(_ballY){
                while (!checkPreUp()){
                        try {
                                _ballY.wait();
                        } catch (InterruptedException e) {
                                Thread.currentThread().interrupt();
                                return;
                        }
            }
                _ballY++;
                _ballY.notifyAll();
        }
}

public void down() throws Exception {
        synchronized(_ballY){
                while (!checkPreDown()){
                        try {
                                _ballY.wait();
                        } catch (InterruptedException e) {
```

```
                                Thread.currentThread().interrupt();
                                return;
                        }
                }
                _ballY--;
                _ballY.notifyAll();
        }
}

public void left() throws Exception {
        synchronized(_ballX){
                while (!checkPreLeft()){
                        try {
                                _ballX.wait();
                        } catch (InterruptedException e) {
                                Thread.currentThread().interrupt();
                                return;
                        }
                }
                _ballX--;
                _ballX.notifyAll();
        }
}

public void right() throws Exception {
        synchronized(_ballX){
                while (!checkPreRight()){
                        try {
                                _ballX.wait();
                        } catch (InterruptedException e) {
                                Thread.currentThread().interrupt();
                                return;
                        }}
                _ballX++;
                _ballX.notifyAll();
        }
    }
}
```

<u>סעיף ג</u>:

נראה תסריט בו יווצר חבק:
נניח שהכדור נמצא בפינה השמאלית התחתונה של המלבן התוחם אותו. כאשר התרד האופקי ירצה להזיזו שמאלה, הוא יגלה כי תנאי-הקדם אינו מתקיים ועל כן יכנס למצב המתנה. כמו כן, כאשר התרד האנכי ינסה לבצע פעולת

הזזה למטה, יווכח גם הוא כי תנאי-הקדם אינו מתקיים ויכנס למצב המתנה. לבסוף, נניח כי תרד הרוח הגריל הזזה למטה, מה שיביא אותו להמתנה. בסה"כ שלושת התרדים נמצאים במצב המתנה, כלומר המערכת נמצאת בחבק.

על מנת למנוע חבקים במערכת, נגביל את משך ההמתנה של תרד ל-10 שניות, אלא שפתרון זה לבדו לא ימנע את החבק. נוסיף למתודה פרמטר אשר יסמן לתרד האם עליו "להתמיד" בנסיונותיו להזיז את הכדור באותו הכיוון, או שמא עליו "להתייאש" ולצאת מהמתודה אף על פי שלא בצע את ההזזה שרצה לעשות. מכיוון שתרד הרוח יכול להגריל כיווני תנועה שונים, ניתן לו להפעיל את פעולות התזוזה כאשר "דגל ההתמדה" שלו הינו שלילי, בעוד יתר התרדים "יתמידו" בפעולותיהם. לכשתגיע המערכת למצב אשר תואר בתחילת התשובה, יחזור תרד הרוח מן הפעולה (בלא שהזיז את הכדור) ויגריל כיוון תנועה כלשהו.

```
class SimpleBlockedBall extends CheckedBlockedBall {

...

public void up(boolean persistencyFlag) throws Exception {
        synchronized (_ballY){
                while (!checkPreUp()){
                        try {
                                _ballY.wait(10000);
                        If (persistencyFlag && !checkPreUp())
                          return;
                        } catch (InterruptedException e) {
                                Thread.currentThread().interrupt();
                                return;
                        }
            }
                _ballY++;
                _ballY.notifyAll();
        }
}

public void down(boolean persistencyFlag) throws Exception {
        synchronized(_ballY){
                while (!checkPreDown()){
                        try {
                                _ballY.wait(10000);
                        If (persistencyFlag && !checkPreUp())
                          return;
                        } catch (InterruptedException e) {
                                Thread.currentThread().interrupt();
                                return;
                        }
            }
                _ballY--;
                _ballY.notifyAll();
```

```
        }
}

public void left(boolean persistencyFlag) throws Exception {
        synchronized(_ballX){
                while (!checkPreLeft()){
                        try {
                                _ballX.wait(10000);
                        If (persistencyFlag && !checkPreUp())
                          return;
                        } catch (InterruptedException e) {
                                Thread.currentThread().interrupt();
                                return;
                        }
            }
                _ballX--;
                _ballX.notifyAll();
        }
}

public void right(boolean persistencyFlag) throws Exception {
        synchronized(_ballX){
                while (!checkPreRight()){
                        try {
                                _ballX.wait(10000);
                        If (persistencyFlag && !checkPreUp())
                          return;
                        } catch (InterruptedException e) {
                                Thread.currentThread().interrupt();
                                return;
                        }}
                _ballX++;
                _ballX.notifyAll();
        }
    }
}
```

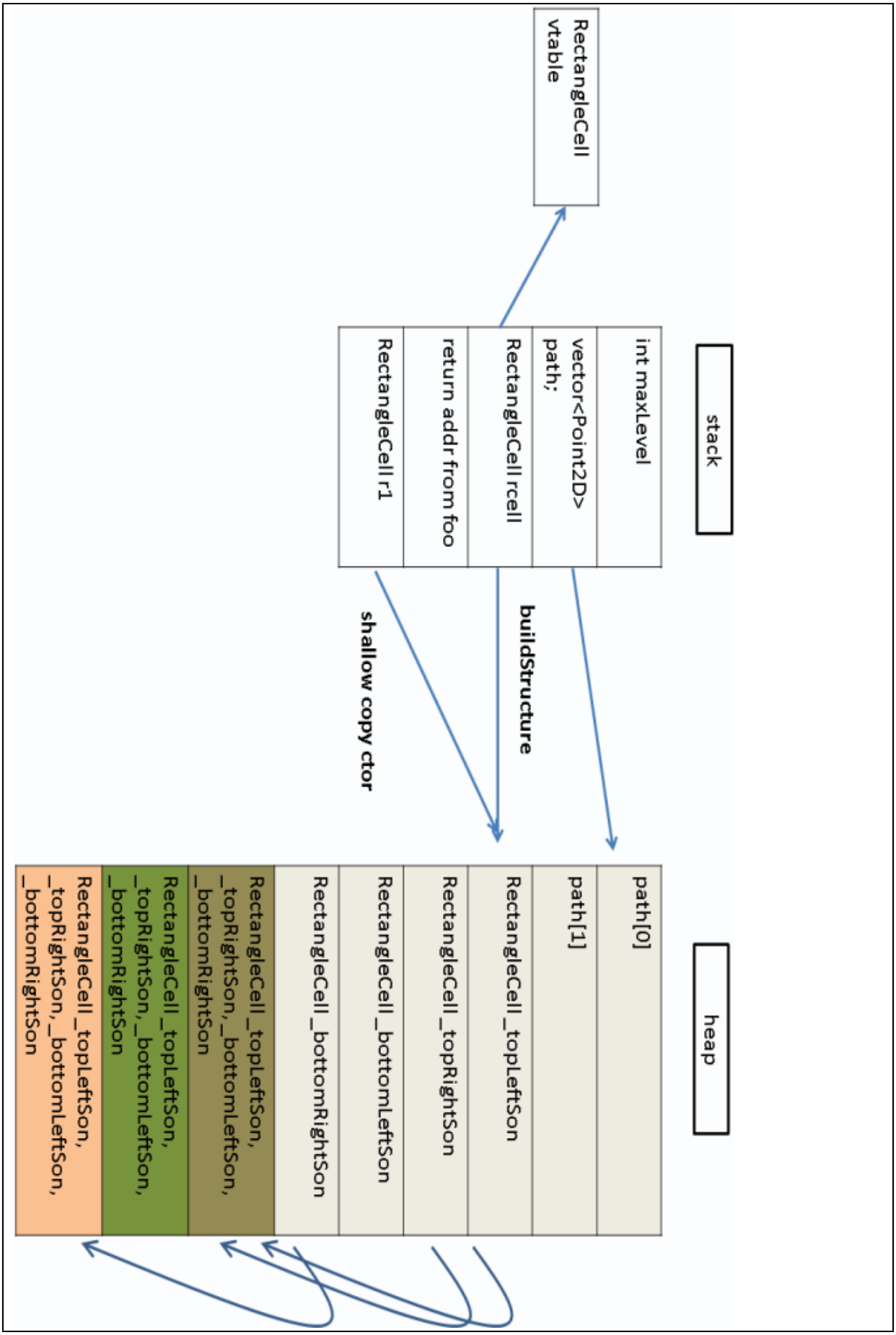
שאלה 2	**(30 נקודות)**

סעיף א (16 נקודות)

```
void RectangleCell::retrieveCells(Point2D center, float radius, vector<SceneCell*> &retrievedCells)
{
	if (!isIntersect(center, radius, *this))
		return;

	if (!_topLeftSon && ! _topRightSon && ! _bottomLeftSon && !_bottomRightSon)
		retrievedCells.push_back(this);

	if (_topLeftSon)
		_topLeftSon->retrieveCells(center, radius, retrievedCells);

	if (_topRightSon)
		_topRightSon->retrieveCells(center, radius, retrievedCells);

	if (_bottomLeftSon)
		_bottomLeftSon->retrieveCells(center, radius, retrievedCells);

	if (_bottomRightSon)
		_bottomRightSon->retrieveCells(center, radius, retrievedCells);
}


void RectangleCell::subdivide(int maxLevel, const Point2D &p)
{
	if (maxLevel=0 || !isIntersect(p, 0, *this))
		return;

	maxLevel --;
	float sizeX = (_right - _left) /2;
	float sizeY = (_top - _bottom) /2;

	if (!_topLeftSon)
		_topLeftSon = new RectangleCell(_left, _bottom + sizeY, _right-sizeX, _top);
	_topLeftSon ->subdivide(maxLevel,p);

	if (!_topRightSon)
		_topRightSon = new RectangleCell(_left+sizeX, _bottom+sizeY, _right, _top);
	_topRightSon ->subdivide(maxLevel,p);

	if (!_bottomLeftSon)
		_bottomLeftSon = new RectangleCell(_left, _bottom, _right-sizeX, _top-sizeY);
```

```
	_bottomLeftSon ->subdivide(maxLevel,p);

	if (!_bottomRightSon)
		_bottomRightSon = new RectangleCell(_left+sizeX, _bottom, _right, _top-sizeY);
	_bottomRightSon ->subdivide(maxLevel,p);
}

void RectangleCell::buildStructure(int maxLevel, std::vector<Point2D> ballPath)
{
	for (std::vector<Point2D>::const_iterator &cit = ballPath.begin(); cit!= ballPath.end(); cit++)
	{
		subdivide(maxLevel, *(cit));
	}
}
```

סעיף ג (14 נקודות)


stack
heap
int maxLevel
vector<Point2D> path;
RectangleCell rcell
return addr from foo
RectangleCell r1
RectangleCell vtable
buildStructure
shallow copy ctor
path[0]
path[1]
RectangleCell_topLeftSon
RectangleCell_topRightSon
RectangleCell_bottomLeftSon
RectangleCell_bottomRightSon
RectangleCell_topLeftSon, _topRightSon, _bottomLeftSon, _bottomRightSon
RectangleCell_topLeftSon, _topRightSon, _bottomLeftSon, _bottomRightSon
RectangleCell_topLeftSon, _topRightSon, _bottomLeftSon, _bottomRightSon

שאלה 3 (10 נקודות)

התרחיש אפשרי **כאשר הת'רדים ב executor מסירים את ה task_ של הלקוח מתור המשימות לפני שהת'רד של ה Reactor מוסיף את אותו task_ בעקבות קריאת בתים נוספים מה socket של הלקוח**. כך שבתור אין אמנם את ה task_ אך הוא נמצא אצל הת'רדים ב executor.
כך שבסופו של דבר, יתכן כי כל הת'רדים ב executor מחזיקים את אותו task_, כאשר רק אחד מהם רץ וכל השאר במצב blocked עקב הסינכרון של מתודת ה ()run.
כל זאת, כאשר ממתין בתור של ה executor, task_של לקוח אחר.

שאלה 4 (20 נקודות)

סעיף א (10 נקודות)

```
import java.io.Serializable;
import java.rmi.RemoteException;
import java.util.HashSet;

interface Reply {
	int getNumericValue();
	String getNumericString();
}

interface ReplyWithData<T extends Serializable> extends Reply {
	T getData();
}

interface IRC  extends java.rmi.Remote {
	Reply nick(String nickName) throws RemoteException;
	Reply user(String userName) throws RemoteException;
	void quit(String quitMessage) throws RemoteException;
	Reply join(String channel) throws RemoteException;
	Reply part(String channel) throws RemoteException;
	ReplyWithData<HashSet<String>> names(String channel) throws RemoteException;
	ReplyWithData<HashSet<String>> list() throws RemoteException;
	Reply kick(String channel) throws RemoteException;
}
```

סעיף ב (10 נקודות)

```
public class IRCServer {
	private final static String IRC_STUB_NAME = "IRC";

	public static void main(String[] args) {
```

```
                String rmiregistryHost = args[0];
                int rmiregistryPort = Integer.parseInt(args[1]);

                IRC irc = new IRCImpl();
                Naming.rebind(rmiregistryHost + ":" + rmiregistryPort + "/" + IRC_STUB_NAME,irc);
        }
}

>rmiregistry 1876
>java IRCServer localhost 1876
```

שאלה 5 (10 נקודות)

סעיף א (4 נקודות)

```
CREATE TABLE doc (
   id INTEGER PRIMARY KEY,
   link VARCHAR(30),
   authorId INTEGER);


CREATE TABLE author (
   id INTEGER PRIMARY KEY,
   name VARCHAR(30),
   birthdate VARCHAR(30),
   address VARCHAR(30));

ALTER TABLE doc
   ADD FOREIGN KEY (authorId)
   REFERENCES author(id);

CREATE TABLE subject (
         id INTEGER PRIMARY KEY,
         subject VARCHAR(30));

CREATE TABLE subject2doc(
         subjectId INTEGER,
         docId INTEGER);

ALTER TABLE subject2doc
   ADD FOREIGN KEY (subjectId)
   REFERENCES subject(id);

ALTER TABLE subject2doc
   ADD FOREIGN KEY (docId)
   REFERENCES doc(id);
```

סעיף ב (6 נקודות)

```
SELECT subject.subject, author.address
FROM (((author join doc on author.id = doc.authorI)
         join subject2doc on doc.Id = subject2doc.docId)
         join subject  on subject.Id = subject2doc.subjectId)
ORDER BY subject.subject, doc.id
```