גיליון תשובות

מספר נבחן:_____

שאלה 1 (25 נקודות)

א.

```
ImageRepository::ImageRepository(int n){
	_capacity = n;
	_free = n;
	_images = new Image*[n];
}

void ImageRepository::Insert(Image *arr, int n){
	for (int i = 0; i <  n; i++)
		_images[_capacity - _free + i] = arr[i].Clone();
	_free -= n;
}

virtual Image::Image* Clone() = 0;
Image* Scenery::Clone() { return  new Scenery(*this); }
Image* Portrait::Clone() {return  new Portrait(*this);     }
```

מפתח:

No New Image*[] -5
No Clone() -5

ב.

```
ImageRepository& operator=(ImageRepository&& rhs) {
   steal(rhs);
   return *this;
}

void steal(ImageRepository& other) {
   _images = other._images;
   _capacity = other._capacity;
   _free = other._free;
```

```
    other._images = 0;
}
```

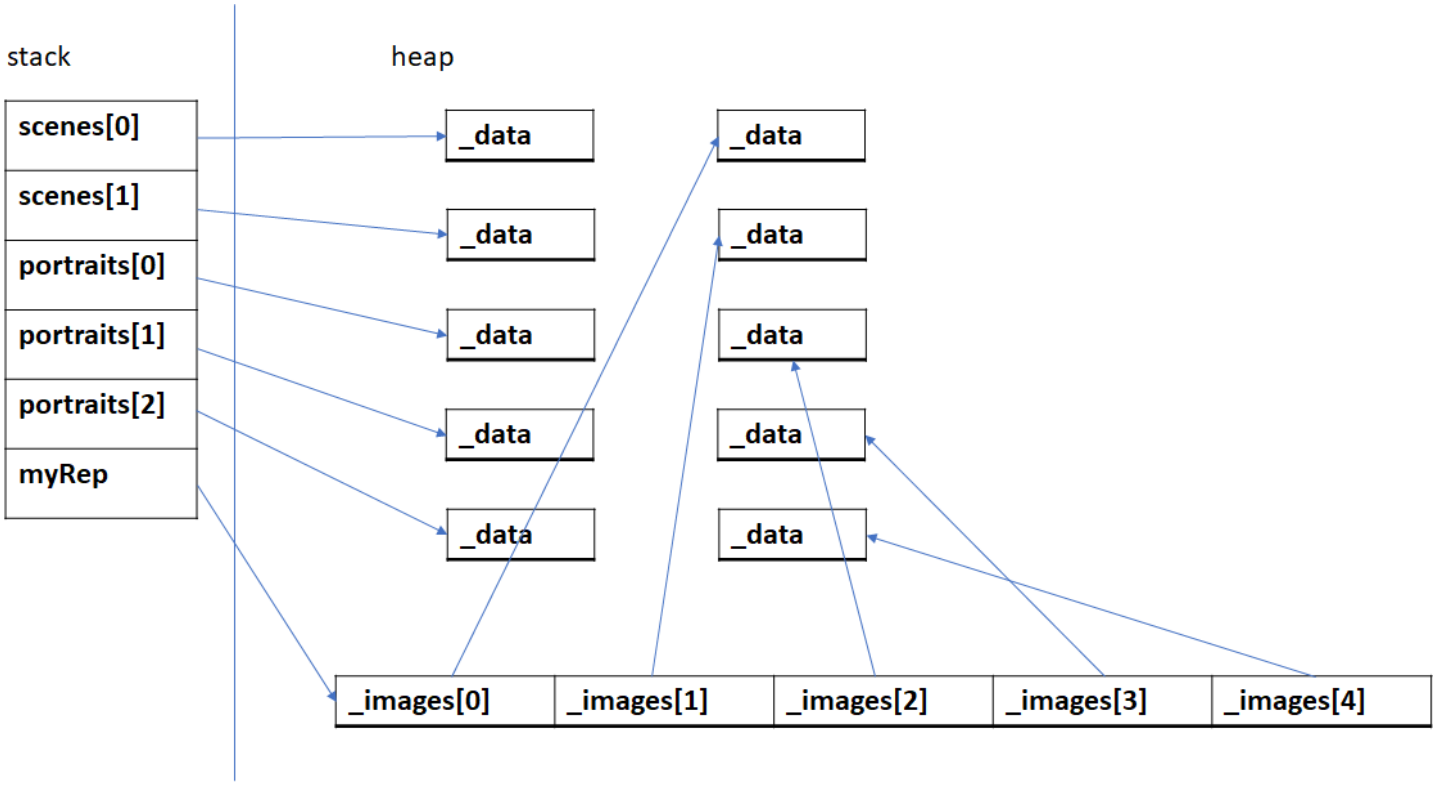
מפתח:

No Move Ass -10

מי ששינה\מחק את קוד בMAIN במקום להוסיף קוד כפי שנכתב בשאלה ("הוסיפו את הנדרש (ואך ורק את הנדרש)") לא קיבל נקודות.

Implementation instead of Move Ass. of: Copy Ass./Move Copy Ctor /no steal -3--7

ג.



מפתח:

Heap, stack -2--3

שאלה 2	(30 נקודות)

א. ההדפסה האפשרית היחידה היא "22"

הסבר: מה שנשלח לבנאי של tasks הוא הכתובת של אובייקט a.
ב main , כאשר אנחנו מקצים אובייקט חדש מסוג A, ומכניסים את כתובתו למשתנה a, זה לא משפיע כלל על הכתובת שאותה קיבלו tasks לפני זה.
לכן, אין לזה השפעה על ההדפסה.
שימו לב: main thread לא מדפיס, רק התרדים האחרים מדפיסים.

ב.1

נדרשים 4 משאבים לפחות.

הסבר: אפילו אם כל תרד יתפוס משאב אחד, עדיין יהיה משאב פנוי (המשאב הרביעי) כך שאחד התרדים יוכל לתפוס שני משאבים ולבצע את עבודתו. בסיום העבודה הוא ישחרר את שני המשאבים שהיו לרשותו, ויאפשר לאחרים לתפוס אותם.

ב.2

נדרש משאב 1 לפחות.

הסבר: תרד כלשהו יתפוס את המשאב ויוכל לבצע את עבודתו. לאחר סיום עבודתו הוא ישחרר את המשאב, ואז יתאפשר לאחרים גם כן לתפוס את המשאב.

ג.1

תנאי סיום: ב threads ישנם שלושה תרדים, כלומר threads.size == 3, וגם מתקיים שרק אחד מהם מנהיג, כלומר countLeaders(threads) ==1

ג.2

תנאי הסיום לא בהכרח מתקיים בכל ריצה בגלל שת'רד יכול להיבחר למנהיג עוד לפני ששלושת התרדים הוגדרו. זה עלול להוביל למצב בו יש כמה מנהיגים.

ג.3

ישנן אפשרויות רבות לתקן קוד. כל אפשרות נכונה התקבלה.
אפשרות הפשוטה ביותר זה להוציא את שורת ההרצה של תרדים מהלולאה ב main, ולהריץ אותם בלולאה נפרדת כאשר הם כבר כולם נוצרו ונכנסו ל queue.

שאלה 3	(30 נקודות)

מומלץ לקרוא בעיון את התשובות לפני שמגישים ערעור

<u>סעיף א</u>:

```
public class Reactor<T> implements Server<T> {

   public static Logger logger = new Logger("reactor.log");
   …

   private void handleAccept(ServerSocketChannel serverChan,
                             Selector selector) throws IOException {
        SocketChannel clientChan = serverChan.accept();
        Reactor.logger.log("connected".getBytes());
        …
   }

   …

}

public class NonBlockingConnectionHandler<T> implements
                                            ConnectionHandler<T> {
…
public Runnable continueRead() {
   …
   if (success) {
      Reactor.logger.log("read".getBytes());
      buf.flip();
      return () -> {
        try {
          while (buf.hasRemaining()) {
            T nextMessage = encdec.decodeNextByte(buf.get());
            if (nextMessage != null) {
              T response = protocol.process(nextMessage);
              Reactor.logger.log("processed".getBytes());
              …
           }
           …
        };
      }
      …
```

```
    }
    ...
    public void continueWrite() {
       while (!writeQueue.isEmpty()) {
          try {
            ByteBuffer top = writeQueue.peek();
            chan.write(top);
            Reactor.logger.log("written".getBytes());
            ...
           }
           ...
       }
       ...
    }
    ...
}
```

סעיף ב

ת'רדים: ת'רד התקשורת (Selector Thread), ת'רד עבודה (Executor Thread)
סיבות: סנכרון המתודה log, כתיבה לקובץ ב i/o blocking

סעיף ג

כדי לקבל את מלוא הנקודות, צריך היה לעמוד בקריטריונים הבאים:

- אף ת'רד לא עובר למצב blocked בשל הדיווח לקובץ הלוג.
- אין מצב בו שני ת'רדים כותבים במקביל לקובץ הלוג.
- כל הבתים נכתבים לקובץ הלוג.
- ניסיון הכתיבה לקובץ הלוג נעשה רק כאשר הקובץ זמין לכתיבה.

מי שלדוגמא רק שינה את הכתיבה ל FileOutputStream לכתיבה ל FileChannel (מקונפג ל non-blocking) ללא סנכרון, קיבל רק 7 נקודות:

- לא מונע כתיבה במקביל של שני ת'רדים שונים (תקשורת ועבודה, או עבודה ועבודה)
- לא מבטיח שכל הבתים ייכתבו (ב non-blocking פעולת write כותבת 0 בתים או יותר, אך לא בהכרח את כל המערך).

במסגרת לימוד קוד הריאקטור, ראינו בהרצאות ובתרגול שני מנגנונים המונעים מעבר למצב blocked:

- שליחת בתים ללקוח ע"י מנגנון ה writeQueue.
- שינוי הרישום מ OP_READ ל OP_READ | OP_WRITE ע"י מנגנון ה sekectorTasks.

המנגנון הנדרש בשאלה זו זהה למנגנון ה writeQueue ודומה למנגנון ה selectorTasks:

- הת'רד היחיד הכותב לקובץ הלוג הוא ת'רד התקשורת.
- ה FileChannel של קובץ הלוג, בקינפוג non-blocking, נרשם בסלקטור עבור אירוע OP_WRITE

- רשימה מוגנת של ByteBuffer מאחסנת את ההודעות לכתיבה בקובץ
- פעולת ה log מוסיפה ByteBuffer לרשימה
- באירוע IsWritable של ה FileChannel, ת'רד התקשורת כותב בתים מהרשימה לקובץ הלוג, כל עוד זה אפשרי.

```
public class Reactor<T> implements Server<T> {
    public static Logger logger;
    …
    public void serve() {
       …
       FileChannel logChan = FileChannel.open("reactor.log");
       logChan.configureBlocking(false);
       logger = new Logger(logChan);
       logChan.register(selector, SelectionKey.OP_WRITE, logger);
       …
    }
    …

    private void handleReadWrite(SelectionKey key) {
      Object handler = key.attachment();
      …
      if (key.isWritable()) {
         if (handler instanceof Logger)
            ((Logger)handler).write();
         else
            ((NonBlockingConnectionHandler<T>)handler).continueWrite();
      }
    }

    …

}

public class Logger {

    private FileChannel chan;
    private Queue<ByteBuffer> logQueue;

    Logger(FileChannel chan) throws IOException {
       this.chan = chan;
       this.logQueue = new ConcurrentLinkedQueue<>();
    }

    public synchronized void log(byte[] msg) throws IOException {
       logQueue.add(ByteBuffer.wrap(msg));
    }
```

```
    public void write() {
      while (!logQueue.isEmpty()) {
        ByteBuffer top = logQueue.peek();
        chan.write(top);
        if (top.hasRemaining())
          return;
        else
          logQueue.remove();
      }
    }
}
```

שאלה 4 (15 נקודות)

א.

```
CREATE TABLE Users (
      id     INT          PRIMARY KEY,
      name   TEXT         NOT NULL
);
CREATE TABLE Movies (
      id                  INT   PRIMARY KEY,
      name                TEXT  NOT NULL,
      description         TEXT,
      isAction            BOOL  NOT NULL,
      isRomance           BOOL  NOT NULL,
      isDrama             BOOL  NOT NULL,
      isKids              BOOL  NOT NULL,
      score               FLOAT NOT NULL,
      num_scorers         INT   NOT NULL
);
CREATE TABLE Movie_watches (
      user_id             INT   NOT NULL,
      movie_id            INT   NOT NULL,
      date                DATE  NOT NULL,
      score               INT,
      review              TEXT,

      FOREIGN KEY(user_id)    REFERENCES Users(id)
```

```
      FOREIGN KEY(movie_id)    REFERENCES Movies(id)
      PRIMARY KEY(user_id, movie_id, date)
);
```

ב.

```
class Movie_watch (object):
      def __init__(self, user_id, movie_id, date, score, review):
            self.user_id = user_id
            self.movie_id = movie_id
            self.date = date
            self.score = score
            self.review = review

class Movie_watches:

def __init__(self, conn):
      self._conn = conn

def insert(self, user_id, name):
      self._conn.cursor().execute("""
            INSERT INTO Movie_watches (user_id,movie_id,date)
             VALUES (?,?,?) """,
             [user_id, movie_id, date])


def update(self, user_id, movie_id, date, score, review):
      self._conn.cursor().execute("""
            UPDATE Movie_watches SET score = (?), review = (?)
             WHERE user_id = (?) AND movie_id = (?) AND date = (?) """,
             [score, review, user_id, movie_id, date,])
```

מי שהשתמשו בפונקציה getDate ב-insert במקום להשתמש ב-date בגלל הtypo שתוקנה קבלו את כל הנקודות.

הרבה התבלבלו בDTO ושמו רק את 3 השדות הראשונים, כי כך הטבלה מאותחלת באופן ראשוני בסיפור בשאלה. שימו לב שDTO הוא אובייקט כללי להחזקת שורה בטבלה (שמכילה באופן כללי גם את score ו-review) ולכן על האובייקט הזה לכלול את כל השדות המתאימים לעבודות הטבלה כולל השדות הללו.

ג.

התקבלו 2 התשובות הבאות:

```
SELECT date, Movies.name, Movie_watches.score, Movie_watches.review FROM Movies INNER JOIN SELECT * FROM Movie_watches WHERE user_id = $id AND YEAR(date) = $year ON Movies.id = Movie_watches.movie_id
```

```
SELECT date, Movies.name, Movie_watches.score, Movie_watches.review FROM Movies INNER JOIN Movie_watches ON Movies.id = Movie_watches.movie_id WHERE user_id = $id AND YEAR(date) = $year
```