מבוא לתקשורת מחשבים
המחלקה למדעי המחשב
פרויקט גמר

הנחיות:

- יש להגיש את הפרויקט בזוגות בלבד
- תאריך הגשה: 13/04/2024 שעה 23:59 בתיבת ההגשה ב-**Moodle**.

במהלך הקורס, עברנו על מספר פרוטוקולים שונים במודל חמש השכבות ברשתות מחשבים.
הסמסטר, תממשו פרוטוקול תקשורת פשוט המזכיר במאפייניו מספר רב של אלמנטים משכבת התעבורה.
מטרת הפרויקט הוא לסמלץ (לדמות) תהליך של שליחת הודעה אחת מהמקלט למשדר, פירוקה למנות מידע, קבלת המנות אצל המקלט, שליחת הודעת **ACK** למשדר והמשך שידור עד קבלת כל הפקטות אצל המקלט.
על מנת לממש סימולציה זו, הנכם מצוידים בקובץ **Python** (סיומת **py**) בשם: **PacketTransmissionProtocolProject.py**
שעליכם לערוך במקומות המתאימים. מוזמנים לכתוב בפייתון באמצעות **IDE** כדוגמת **PyCharm** (חינמי בגרסת ה-**Community** וחינמי לסטודנטים בגרסת ה-**Enterprise**).

הפרויקט פורק עבורכם למשימות המתמקדות בטיפול במחלקות ספציפיות ולתתי משימות המתמקדות בשיטות מסויימות בתוך כל מחלקה.

משימה 1: מימוש המחלקה המייצגת מנת מידע (Packet)

מחלקה זו אחראית לייצוג מנות המידע שישלחו מהמשדר למקלט ולהפך.

משימה 1.1: מימוש הבנאי של המחלקה Packet

לשם כך, ממשו תחילה את המחלקה (**class**) בשם **Packet**.
הבנאי של **Packet** מקבל את הפרמטרים הבאים:

א. **source_address** – כתובת ה-**IP** של המשדר (טיפוס **string**).
ב. **destination_address** – כתובת ה-**IP** של המקלט (מטיפוס **string**).
ג. **sequence_number** – מספר הפקטה מתוך הרצף הכולל של הפקטות (מטיפוס **int**).
ד. **is_ ack** – יספק אינדקציה האם מדובר בפקטה של המשדר או המקלט (מטיפוס בוליאני).
ה. **data** – שדה המחזיק את המידע עצמו (מטיפוס מחרוזת).

עליכם לממש את המטודה _ **_init**_ (הבנאי) של מחלקת ה-**Packet**. הבנאי מקבל את הערכים המתוארים בחתימתו ושומר את אותם הערכים באמצעות משתנה המחלקה הנשמרים באותו השם (לא לשכוח את **self**).
תזכורת: שדות מחלקה נדרשים להיות **private** באמצעות שני קווים תחתונים.

דוגמה:

self._ _id = id

חתימת הבנאי (שורה 2-3):

def _ _init_ _(self, source_address, destination_address, sequence_number, is_ack=False, data=None):

משימה 1.2: מימוש השיטה repr של המחלקה Packet:

השיטה ה-_ _**repr**_ _ מחזירה יצוג מחרוזתי קריא לבני אדם של אובייקט מסוים.
מימוש של מטודה זו תעזור לכם לצפות בפרטי החבילה כאשר תפעילו את הפונקציה **print** על אובייקט מסוג **Packet**.
על המטודה להחזיר מחרוזת המורכבת מפרטי החבילה בצורה הבאה:

"Packet(Source IP: A, Dest IP: B, #Seq: C, Is ACK: D, Data: E)"

כאשר **A-E** הם הערכים המתאימים של האובייקט **Packet**. ישנו רווח אחד בלבד בין כל מחרוזת לערך מהפקטה.

חתימת השיטה (שורה 7):

def _ _repr_ _(self):

משימה 1.3: מימוש getter **ו-**setters **עבור השדות של** Packet**:**

בחלק זה עליכם לממש את המטודות הבאות:

```
def get_source_address(self):                # line 11
def get_destination_address(self):           # line 15
def get_sequence_number(self):               # line 19
def set_sequence_number(self, seq_num):      # line 23
def get_is_ack(self):                        # line 27
def get_data(self):                          # line 31
```

תזכורת: שיטות **getters** נדרשות להחזיר את ערכי השדות של המחלקה בעוד ש-**setters** רק מעדכנות את הערך החדש המתקבל בחתימת השיטה.

<u>משימה 2: מימוש המחלקה Communicator</u>
ישנה פונקציונליות זהה גם אצל המשדר וגם אצל המקלט.
עובדה זו דורשת מאיתנו לממש את המחלקה **Communicator** שתחזיק ערכים ושיטות המשותפות גם למשדר וגם למקלט.

<u>משימה 2.1: מימוש הבנאי של המחלקה Communicator</u>
הבנאי של **Communicator** מקבל את הפרמטר הבא:
א. **address** – כתובת ה-**IP** של המשדר או המקלט (טיפוס **string**).

בנוסף, מחזיק ה-**Communicator** שדה פרטי המציין את מספר הפקטה העכשווית שהוא נדרש לשלוח או לקבל (**self._ _current_seq_num**) . יש לאתחלה ל-**None** בהתחלה.

<u>חתימת הבנאי (שורה 36)</u>:
def _ _init_ _(self, address):

<u>משימה 2.2: מימוש getter ו-setter של שדות המחלקה Communicator</u>
ממשו את השיטות הבאות. השיטות פשוט עושות **get** לשדה הנתון או **set** בהינתן פרמטר לשדה המטרה.
def get_address(self): **#line 40**
def get_current_sequence_number(self): **#line 44**
def set_current_sequence_number(self, seq_num): #line 48

<u>משימה 2.3: מימוש שליחת פקטה</u>
פונקציה זו מדפיסה את ההודעה הבאה לדוגמה:
Sender: Packet Seq Num: 0 was sent
על הפונקציה להדפיס את הנאמר ולהחזיר את אותה ה-**packet**.
<u>חתימת השיטה</u>:
def send_packet(self, packet): #line 52

<u>משימה 2.4: עדכון מספר הפקטה העכשווית</u>
שיטה זו אחראית לעדכון מספר הפקטה (**sequence number**) ב-1 שיש לשלוח במקרה של קבלת **Ack** בצורה תקינה.

<u>חתימת השיטה</u>:
def increment_current_seq_num(self): #line 56

משימה 3: מימוש המחלקה Sender

בסעיף זה, עליכם לממש את המחלקה **Sender** המייצגת את המשדר (הגורם השולח את הפקטות).

משימה 3.1: מימוש בנאי המחלקה Sender:

ממשו במחלקת **Sender**, את הבנאי הכולל בחתימתו את השדה הבא:

א. **address** – כתובת ה-**IP** של המשדר (טיפוס **string**).

ב. **num_letters_in_packet** – שדה המחזיק כמה אותיות ניתן לשמור בפקטה אחת.

def __init__(self, address, num_letters_in_packet): #line 61

במימושכם, השתמשו בבנאי של **Communicator** שכבר מימשתם.

משימה 3.2: מימוש הכנת הפקטות לשליחה אצל המשדר

ממשו את השיטה הבאה בשם:

def prepare_packets(self, message, destination_address): # line 65

השיטה מקבלת את ההודעה הנדרשת להשלח ומחזירה **N** פקטות המרכיבות את ההודעה.

תחילה עליכם לחלק את הודעת הקלט ל-**N** פקטות כך שכל פקטה מכילה **X** אותיות מההודעה. בקוד המצורף, הצבנו **X=3** אותיות.

דוגמה:

עבור ההודעה "**What is up?**" של השולח

תוכן הפקטות יהיו משמאל לימין:

["Wha", "t i", "s u", "p? "]

שימו לב שכל פקטה צריכה להכיל בדיוק 3 אותיות. יש לדאוג להוסיף רווחים על מנת להשלים לפקטה אחרונה בת 3 אותיות כמו היתר.

השתמשו במחלקת **Packet** על מנת ליצור את הפקטות.

על מנת להקל, ה-**sequence number** של כל פקטה הוא מיקומה ברשימת הפקטות הנדרשות להשלח (החל מהמספר 0 ואילך).

שמרו את אותן פקטות ברשימה בשם **packets**. לבסוף, עליכם להחזיר רשימה זו.

משימה 3.3: מימוש קבלת ACK מהמקלט

ממשו את השיטה הבאה בשם:

def receive_ack(self, acknowledgment_packet): # line 69

שיטה זו מקבלת את הפקטה ופשוט בודקת האם מדובר בהודעת **ACK**. אם אכן מדובר בהודעת **ACK** מחזירה השיטה **True**, אחרת **False**.

<u>משימה 4: מימוש המחלקה Receiver</u>

בסעיף זה, עליכם לממש את המחלקה **Receiver** המייצגת את המקלט (הגורם המקבל את הפקטות).

<u>משימה 4.1: מימוש בנאי המחלקה Receiver:</u>

ממשו במחלקת **Sender**, את הבנאי הכולל בחתימתו את השדה הבא:

א. **address** – כתובת ה-**IP** של המשדר (טיפוס **string**).

def __init__(self, address): #line 74

במימושכם, השתמשו בבנאי של **Communicator** שמימשתם.

בנוסף, אתחלו רשימה ריקה כשדרה פרטי של **Receiver** בשם **received_packets**.

<u>משימה 4.2: מימוש השיטה receive_packet:</u>

שיטה זו אחראית לטיפול בפקטה הנשלחת מהמשדר והמתקבלת אצל המקלט.

השיטה מקבלת **packet** ומחזירה **ack**.

עליכם לשמור את הפקטה המתקבלת בתוך רשימת הפקטות שהתקבלו אצל ה-**Receiver**.

בנוסף, עליכם ליצור **Acknowledgment** אותו תשלחו בהמשך ל-**Sender**.

ה-**Acknowledgement** הוא **Packet** כאשר המקור והיעד מתחלפים ובעל אותו **sequence number** כמו הפקטה המקורית שהתקבלה. ההודעה עצמה היא מסוג **ACK** וללא תוכן.

הדפיסו הודעה למסך במקרה של קבלה (לדוגמה):

Receiver: Received packet seq num: 2

<u>חתימת השיטה</u>:

def receive_packet(self, packet): #line 78

<u>משימה 4.3: מימוש השיטה get_message_by_received_packets:</u>

שיטה זו לוקחת את כל ההודעות שהתקבלו אצל המקלט ובונה את ההודעה המקורית ע"י איחוד של כל **N** אותיות בכל פקטה לידי הודעה אחת. השיטה מחזירה לבסוף את ההודעה שנבנתה.

def get_message_by_received_packets(self): #line 82

הסבר על קוד ה-Main
את קוד ה-**Main** המטפל בהרצה עצמה (החל משורה 86) והמתחיל כך:
if __name__ == '__main__':
הנכם מתבקשים לא לשנות את הקוד החל משורה 92 ואילך.
עם זאת, ניתן לשנות את הערכים של המשתנים בשורה 87 עד 90, אבל לא את המשתנים עצמם.

בשורה 92 ו-93 אחראיות לאיתחול **Sender** ו-**Receiver**.
בשורה 95, ה-**Sender** מכין את הפקטות לשליחה.
שורות 97-101 אחראיות להשמת הפקטה הנדרשת להשלח או להתקבל אצל המשדר והמקלט.
שורה 103 אחראית לשמירת ה-**sequence number** של הפקטה האחרונה שכן אנו חייבים תנאי עצירה.

בשורה 110, שולח המשדר את הפקטה.
בשורה 112, מקבל המקלט את הפקטה ומחזירה עבורה פקטת **Ack** מתאימה.
בשורה 114, מקבל המשדר את הודעת ה-**Ack** ובוחן אותה.
בהינתן שההודעה תקינה, מעדכנים המשדר והמקלט את הפקטה הבאה עליהם להעביר ביניהם.
לבסוף, בשורה 123-124 המקלט בונה את ההודעה מאוסף הפקטות שצבר ומדפיס אותם למשתמש.

הנחיות נוספות:
ספקו הערות ברורות ותמציתיות עבור כל שיטה וקטע קוד שלכם.
תעדו כל הנחות או בחירות עיצוב שנעשו על ידיכם במהלך היישום.
פעלו לפי נוהלי קידוד נאותים, כולל שמות משתנים ופונקציות משמעותיים.
השתמשו בהערות כדי להסביר את המטרה והפונקציונליות של מקטעי קוד קריטיים.
ודא שהקוד מאורגן היטב וקל להבנה עבור מישהו שסוקר אותו.

הגשה:
ערכו את שם הקובץ והוסיפו את תעודות הזהות שלכם עם קווים תחתונים מפרידים ביניכם. לדוגמה:
PacketTransmissionProtocolProject_ID1_ID2.py
כאשר **ID1** ו-**ID2** הן תעודות הזהות של שני השותפים.
לבסוף, העלו את קובץ הפייתון למודל.

בהצלחה!!!!!!