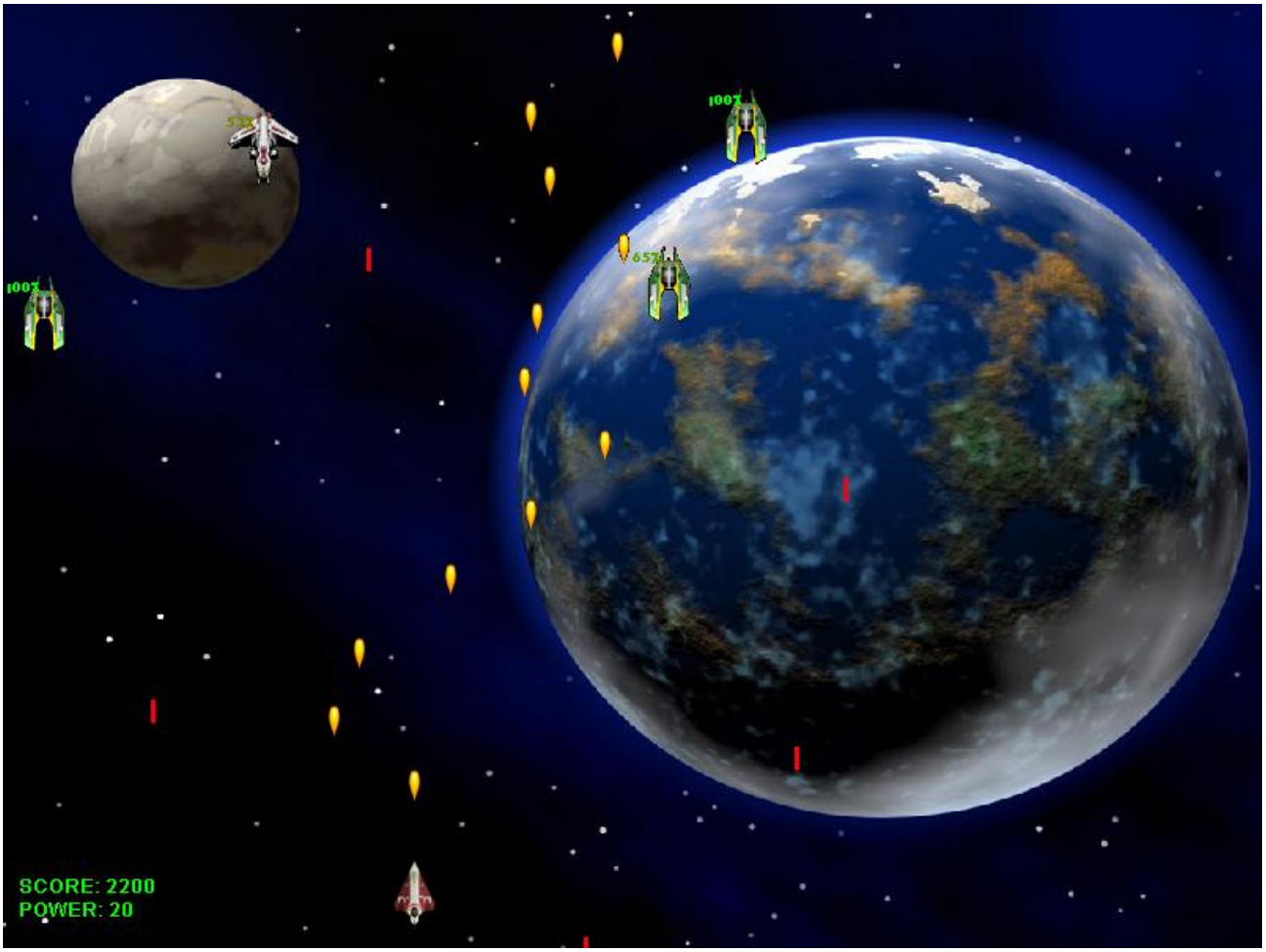
סדנה בתכנות מתקדם בשפת JAVA
פרוייקט מסכם



תאריך: 01.05.2005
מגיש: יוסי שאול
ת.ז. : 032870115

תוכן העניינים

מבוא 3
הקדמה 3
תפריט המשחק 4
ריצת המשחק 7
עצמים דינמיים במשחק 13
משחק רשת 17
תקשורת מחוץ למשחק 17
תקשורת בזמן המשחק 19
צד השרת 22
JMS 22
EJBs 22
בסיס נתונים 23
הוראות התקנה והרצה 24
דרישות המשחק 24
התקנת השרת 24
הרצת המשחק 27
ביבליוגרפיה 29

1 מבוא

כפרוייקט מסכם כתבתי משחק פעולה חללי המאפשר משחק של שחקן יחיד או של שניים דרך הרשת.

המשחק, בסגנון משחקי חלליות של שנות ה-80, מאפשר לשחקן לשלוט בחללית ולהלחם בחלליות של האויב, לעלות שלבים ולשפר את החללית שלו וכלי הנשק שלה. במשחק רשת שני שחקנים יכולים להלחם יחדיו נגד המחשב.

בפרוייקט נעשה שימוש נרחב ביכולות מתקדמות של J2SE ושל J2EE כגון Java 2D API, Java Sound API, JDBC, EJB, Java API for XML (JAXP), JMS ועוד. בכתיבת הפרוייקט הוקפד השימוש בעקרונות תכנות מונחה עצמים.

2 הקדמה

במסמך זה אפרט את החלקים המרכזיים בפרוייקט החל מהעיצוב הכללי של התוכנית ומהלך ריצתה עד לתיאור כמה מהחבילות והיררכיית המחלקות החשובות. תיעוד מלא ניתן למצוא בתוך הקוד או בקבצי תעוד שהופקו בעזרת הכלי Javadoc.

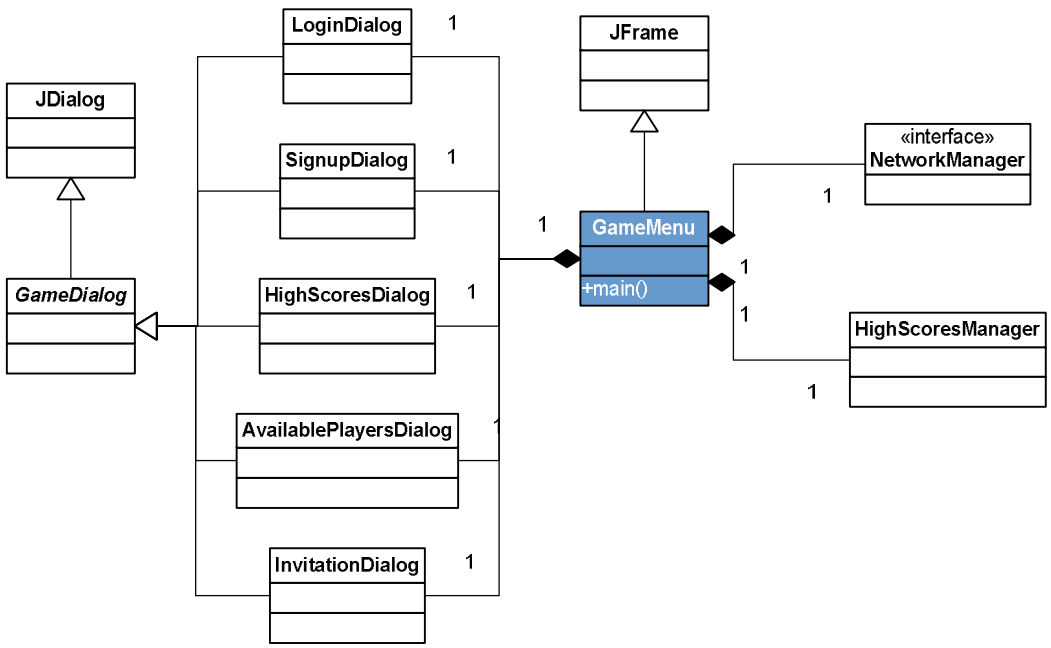
כמו כן יפורטו הרכיבים שבצד השרת, הוראות התקנה והרצה ורשימת מקורות.

3 תפריט המשחק

בעת הפעלת התוכנית יראה המשתמש את תפריט המשחק (איור 3.1). תפריט המשחק הוא החלק המרכזי הראשון של התוכנית (החלק השני, החשוב יותר, הוא הרצת המשחק עצמו) והוא כולל אפשרויות להתחלת משחק חדש, כניסה לשרת המשחק, הרשמה לשרת וסיום התוכנית.

איור 3.1 – תפריט המשחק

המחלקה האחראית לתפריט היא **GameMenu**. באיור 3.2 מוצגת דיאגרמה מחלקה של **GameMenu** ומרכיביה העיקריים.



איור 3.2 – מחלקות עקריות הקשורות ל – GameMenu

NetworkManager:

ממשק זה משמש את ה – **GameMenu** להתקשרות עם שרת המשחק בשלב זה לפני תחילת המשחק (כניסה למערכת, הרשמה שליחת וקבלת הזמנות למשחק רשת וכו'). על המימוש ודרכי הפעולה נרחיב בהמשך בפרק העוסק בתקשורת.

HighScoresManager:

מחלקה זו מנהלת ברמה הלוגית את התוצאות הטובות ביותר שהושגו במשחק. המחלקה כוללת מיון התוצאות הטובות ביותר, שמירה וטעינה מקובץ והשגת שתוצאות הטובות ביותר מן השרת (בעזרת ממשק הרשת).

לחיצה על כפתורי התפריט תפתח בד"כ דיאלוג המראה אינפורמציה (כגון תוצאות טובות ביותר) או מבקש קלט מן המשתמש. כל התפריטים בשלב תפריט המשחק יורשים מן המחלקה **GameDialog** שבתורה יורשת מהרכיב **JDialog** של Swing.

התפריטים השונים בשלב זה הם:

LoginDialog:

כפי שמרמז השם, תפריט זה נועד להתחברות לשרת המשחק ורישום המשתמש כשחקן פנוי. בכדי לפתוח תפריט זה על השחקן ללחוץ על כפתור Login. לאחר שהמשתמש מתחבר למערכת כפתור ה – Login הופך ל – Logout ולחיצה עליו תנתק את המשתמש מהשרת.

SignupDialog:

תפריט זה משמש לרישום משתמש חדש למערכת ומבצע כניסה מיידית לאחר מכן.

HighScoresDialog :

תפריט זה נועד להצגת תוצאות גבוהות ביותר מהקובץ המקומי או מן השרת.

AvailablePlayersDialog :

בכדי להתחיל משחק רשת על השחקן להיות מחובר לשרת (לאחר שנכנס עם שם משתמש וסיסמא) ואז עליו לבחור שותף למשחק ולשלוח לו הזמנה לשחק. את רשימת השחקנים הפנויים ושליחת ההזמנות מבצע המשתמש דרך תפריט זה פשוט ע"י בחירת המשתמש ולחיצה על שליחת הזמנה.

InvitationDialog :

תפריט זה יצוץ כאשר המשתמש קיבל הזמנה לשחק משחק דרך הרשת. בתפריט פשוט זה יכול המשתמש לקבל את ההזמנה או לדחות אותה.

4 ריצת המשחק

כאשר המשתמש מתחיל משחק חדש ה – **GameMenu** יוצר אובייקט **GameLoop** חדש ותהליכון המריץ אובייקט זה. התהליכון של **GameMenu** מצטרף לתהליכון החדש (באמצעות הפונקציה Thread.join()) וממתין עד אשר **GameLoop** יסיים את ריצתו.

המחלקה **GameLoop** מכילה את ליבת המשחק, לולאה הרצה בקצב שנקבע מראש (הקובע את ה - frame rate) עד אשר המשחק מסתיים. בלולאה זו מתבצעות כל הפעולות הנחוצות לאותו ה – frame. פעולות אלו מתחלקות לשלוש קבוצות עקריות: איסוף קלט מהמשתמש (דרך המקלדת או מהרשת), עדכון האובייקטים במשחק ביחס לזמן שעבר מהעדכון הקודם וציור המצב הנוכחי למסך. קטע הקוד של הלולאה מוצג באיור 4.1.

מלבד הלולאה הראשית במחלקה **GameLoop** מאתחלים את רוב מחלקות הניהול העקריות במשחק ומבצעים את המעברים ממצב מסוים במשחק (**GameState**) למצב אחר כאשר זה מסתיים.

איור 4.2 מציג דיאגרמת מחלקה של **GameLoop** הכוללת את המחלקות העקריות שקשורות למחלקה **GameLoop.**

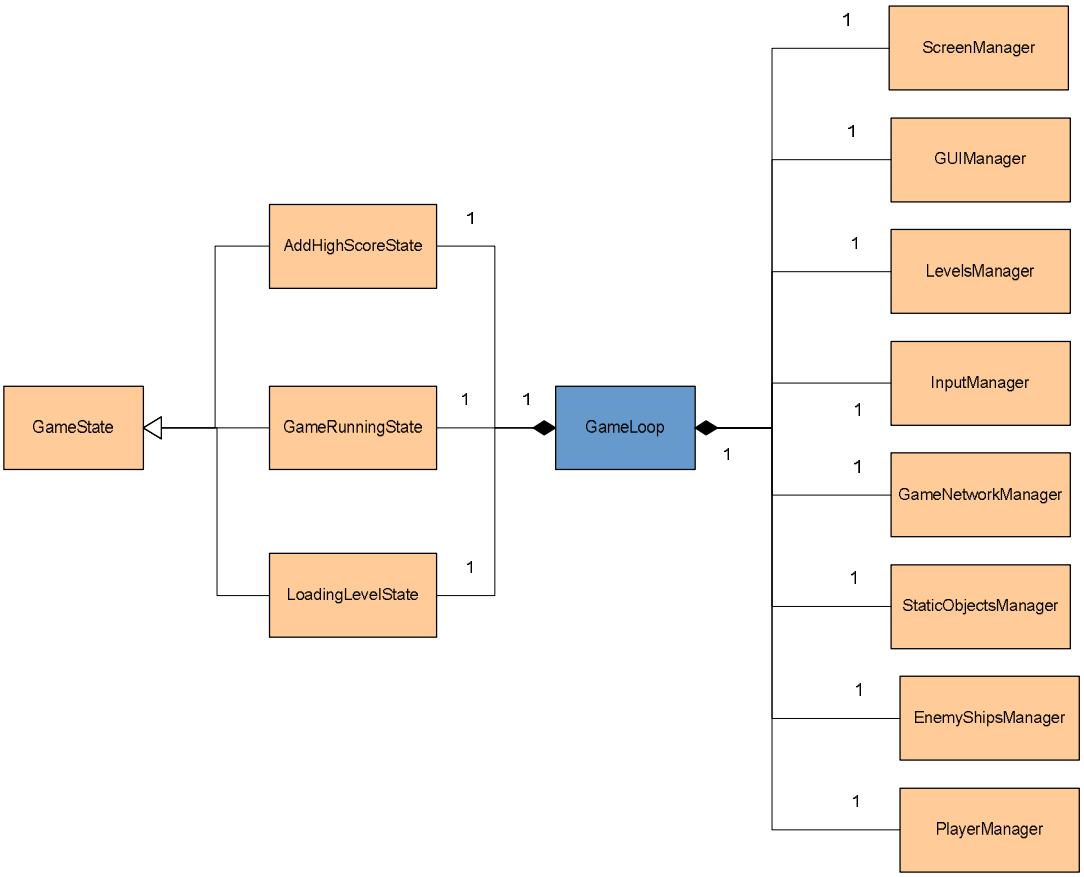
בחלק הימני של האיור מוצגות המחלקות ה"מנהלות" (אלו המסתיימות בשם Manager), כל מחלקה כזו מנהלת את התחום שנקבע לה, כגון קלט, רשת, מסך וכו'. נעבור בקצרה על כל אחת מהמחלקות בהמשך. מן הצד השני ישנן שלוש מחלקות מהטיפוס **GameState**. הממשק **GameState** מגדיר את המתודות שעל מצב מסוים במשחק לממש.

```java
public void run() {

    long prevFrameTime = System.currentTimeMillis();
    curGameState.start();

    while(!inputManager.isQuit()) {

        long currFrameTime = System.currentTimeMillis();

        try {
            Thread.sleep(GameConstants.SLEEP_TIME);
        }
        catch (InterruptedException ie) {
            //
        }

        long elapsedTime = currFrameTime - prevFrameTime;
        prevFrameTime = currFrameTime;

        curGameState.gatherInput(this, elapsedTime);
        curGameState.update(this, elapsedTime);
        curGameState.render(this);

        if (curGameState.isFinished()) {
            changeGameState();
        }

    }

    finalizeGame();

}   // end method run
```

איור 4.1 – מתודת run של המחלקה GameLoop

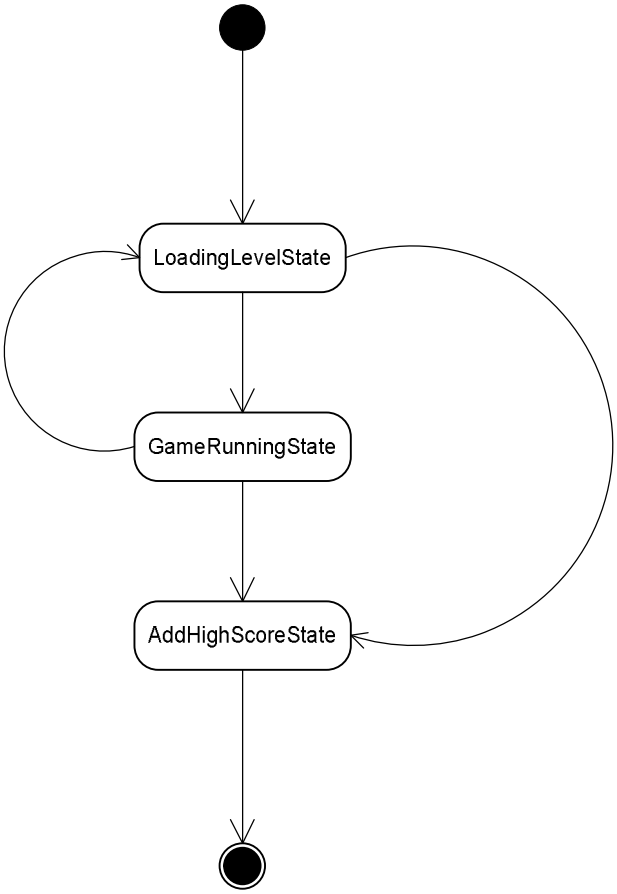
GameState בעצם מטפל במצב ייחודי ונפרד במשחק כמו טעינת שלב, ריצת המשחק, קליטה והכנסת תוצאות טובות ביותר. למה בעצם אנו צריכים להשתמש ב – **GameState**? כפי שהוסבר קודם לכן, במחלקה **GameLoop** רצים בלולאה שבכל איטרציה אוספים קלט, מעדכנים מצב ומציירים למסך. כל אחת מהפעולות הללו תלויה במצב בו אנו נמצאים. לדוגמא, כאשר אנו במצב של טעינת השלב הנוכחי במשחק נרצה לצייר את הרקע והודעה האומרת שהשלב בשלבי טעינה אולם כאשר אנו במצב שבו המשחק רץ נרצה לצייר דברים שונים כגון חלליות האוייב, יריות ועוד. השוני קיים גם בטיפול שונה בקלט ובעידכון המצב הנוכחי והאובייקטים הקשורים אליו.

ע"י שימוש ב – **GameStates** שונים, כל מצב מכמס את הלוגיקה הספציפית לו והטיפול הייחודי של כל מצב בקלט, בעדכון המצב ובציור פשוט וברור יותר. בסוף כל איטרציה ב – **GameLoop** בודקים אם המצב הנוכחי הסתיים ואם כן מבקשים ממנו את המצב הבא ועוברים אליו.



איור 4.2 – דיאגרמת מחלקה של GameLoop

באיור 4.3 מוצגת דיאגרמת מעברים בין המצבים השונים במשחק ובעזרתה נסביר מהם מצבי המשחק השונים ומה תפקידו של כל מצב.



איור 4.3 – דיאגרמת מעברים בין מצבי המשחק השונים

את המשחק אנו מתחילים ב - **LoadingLevelState** האחראי על טעינת השלב הבא מתוך קובץ ה – XML של השלבים וממידע שנשלח ברשת במקרה של משחק רשת. במצב זה טוענים את חלליות האויב, את תמונת הרקע ומבצעים מספר פעולות נוספות הנחוצות להתחלת שלב חדש. אם השחקן עדיין לא סיים את השלב האחרון אנו עוברים למצב **GameRunningState** אחרת עוברים ל – **AddHighScoreState**. המצב **GameRunningState** הוא המצב שבו נמצא רוב הזמן ובו מתבצעת הלוגיקה המרכזית של המשחק. במצב זה השחקן למעשה משחק ושולט בחללית שלו והמחשב שולט ומנווט את חלליות האויב. ממצב זה נצא חזרה ל – **LoadingLevelState** כאשר נסיים שלב או ל – **AddHighScoreState** כאשר החללית של השחקן תושמד ויסתיים המשחק. המצב **AddHighScoreState** הוא הפשוט ביותר ובו בודקים אם השחקן עשה תוצאה שיכולה להכנס לרשימת עשר התוצאות הטובות ביותר (אם כן מבקשים את שמו) ומציעים לשחקן לשלוח את התוצאה לשרת. עם סיום מצב זה נסיים את המשחק ונחזור לתפריט המשחק.

נתאר כעת בקצרה את המחלקות ה"מנהלות" שבהן משתמשים ה- **GameLoop** ומצבי המשחק השונים.

:**ScreenManager**

מחלקה זו אחראית על ניהול המסך (מעבר למצב מסך מלא ויציאה ממנו), משטח הציור, מניעת ריצוד ופעולות נוספות הקשורות לגרפיקה ולמסך. במחלקה זו מוחזק משטח הציור המרכזי (אובייקט מסוג JFrame) שעליו מציירים את כל האובייקטים במשחק. בדרך כלל המתודה *render* של **GameState** תתחיל בהשגת אובייקט ה – **Graphics** מה – **ScreenManager** ותסתיים בקריאה למתודה *show* של **ScreenManager** בכדי להציג את כל שצוייר על המסך.

:**GUIManager**

מחלקה זו משמשת כמחלקת עזר להוספת והצגת רכיבים של **Swing** כגון דיאלוגים שבהם נשתמש במשחק ותיבות טקסט. ישנן מספר בעיות עם הצגת רכיבי **Swing** כאשר משתמשים ב – active rendering כפי שנעשה במשחק (בעיקר ההתעקשות של רכיבי ה – swing לבצע repaint ונעילות כאשר משתמשים בתיבות טקסט), ובמחלקה זו מתמודדים בצורה אחידה עם בעיות אלו.

:**InputManager**

במחלקה זו נאסף הקלט של המשתמש מן המקלדת. מחלקה זו מאזינה לארועי מקלדת המתרחשים ע"ג משטח הציור הרשאי של המשחק ושומרת מצבי מקשים רלוונטיים לאיסוף מאוחר יותר ע"י מצב המשחק הנוכחי.

:**StaticObjectsManager**

במחלקה זו מנהלים ומציירים אובייקטים סטטיים במשחק. אובייקטים לדוגמא הם תמונת ברקע של המשחק וניקוד השחקן.

EnemyShipsManager:

ניהול חלליות האוייב והאובייקטים הדינמיים שהן מייצרות (יריות, בונוסים) מתבצע במחלקה זו. החלליות הן עצמאיות למדי אולם את הקריאות לעדכונן, לבדיקת התנגשויות, לצייר את עצמן וקריאות נוספות מבצעת מחלקה זו. מחלקה זו מממשת את הממשק **ShipContainer** המספק לחלליות שבמנוהלות ע"י מחלקה זו שירותים שונים כגון גישה למנהל הרשת ולאובייקטים אחרים במשחק.

PlayerShipManager:

ניהול חללית השחקן וחללית שחקן הרשת מתבצע המחלקה זו. דרכה מעדכנים את החללית בהתאם לקלט שהכניס המשתמש ולזמן שעבר. גם מחלקה זו מממשת את הממשק **ShipContainer**.

LevelsManager:

מחלקה זו עובדת האופן הדוק עם המצב **LoadingLevelState** ובה למעשה טוענים את פרטי השלב מקובץ XML (הקובץ נקרא level.xml והוא נמצא תחת התיקייה config). את נתוני השלבים השונים במשחק כגון מספר השלב האחרון, מספר וסוגי החלליות בכל שלב, תמונת הרקע בשלב, עורכים בקובץ levels.xml וממנו קוראת מחלקה זו את הנתונים ומעבדת אותם.

GameNetworkManager:

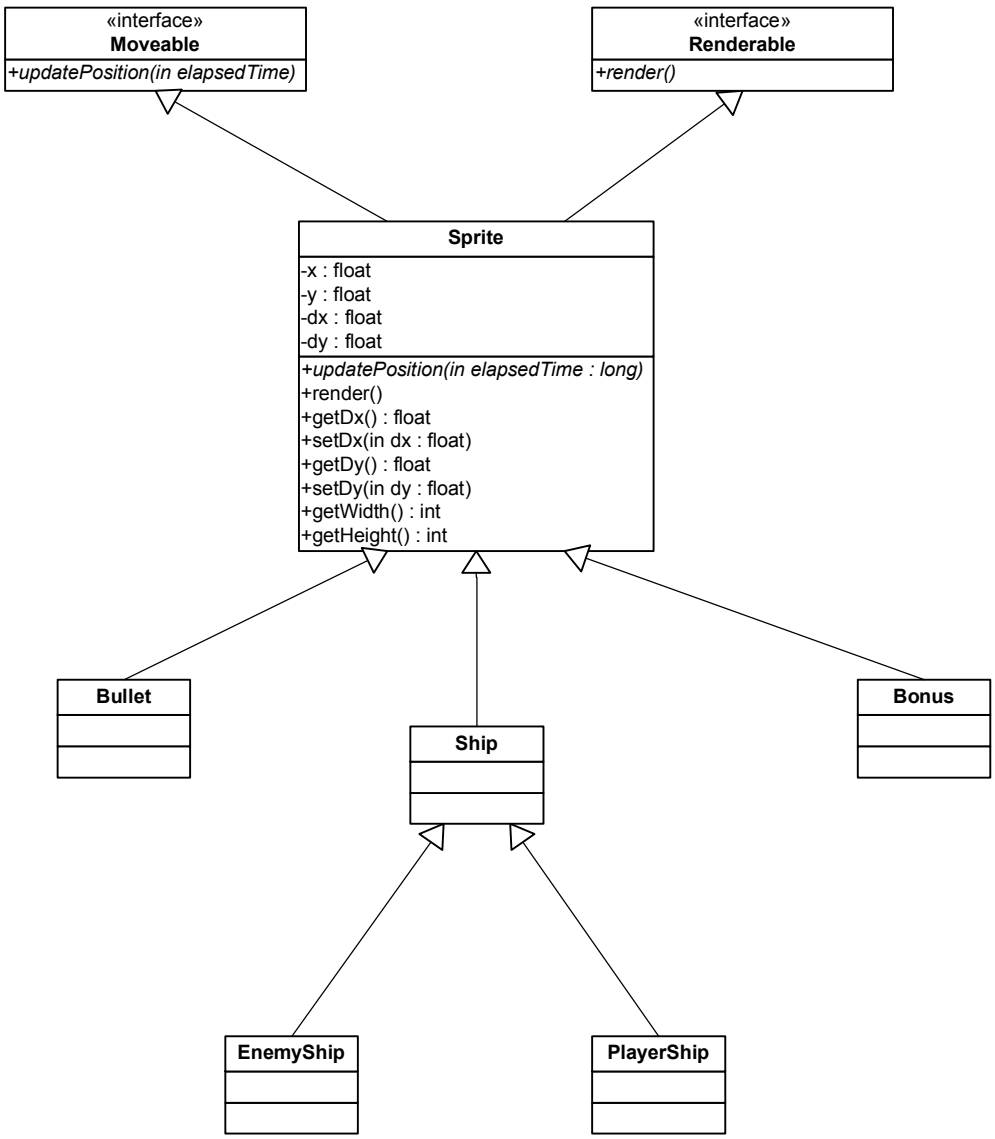
ניהול תעבורת הרשת במהלך המשחק (ולא בשלב תפריט המשחק) מתבצע דרך מחלקה הממממשת את הממשק **GameNetworkManager**. בפרק העוסק ברישות במשחק נפרט יותר על ממשק זה.

5 עצמים דינאמיים במשחק

העצמים הדינמיים במשחק הם העצמים הנעים ופועלים ע"ג המסך בזמן המשחק. העצמים הדינמיים במשחק הם החלליות, היריות הנורות מן החלליות והבונוסים שמשאירות חלק מהחלליות המושמדות.

כל העצמים הללו מרחיבים את המחלקה המופשטת **Sprite**. במונח Sprite מרבים להשתמש בעולם ההדמייה ומשחקי המחשב ומשמעות המונח היא עצם ה"יודע" לנוע, יש לו תכונות כגון מיקום ומהירות והוא יודע לצייר את עצמו.

איור 5.1 מציג את היררכיית העצמים הדינמיים במשחק.



איור 5.1 – עצמים דינמיים במשחק

כפי שניתן לראות ההיררכיה מתחילה בשני ממשקים, **Moveable** המגדיר מתודה אחת, **updatePosition** , המקבלת כפרמטר את הזמן שעבר מן העדכון האחרון (במילישניות) והממשק **Renderable** המגדיר גם הוא מתודה אחת, **render**, המקבלת פרמטר מסוג Graphics באמצעותו מצייר האובייקט את עצמו.

את שני ממשקים אלו מרחיבה וממשת המחלקה המופשטת **Sprite**. המימוש של שני הממשקים הוא די פשוט במחלקה זו. במתודה updatePosition אנו מעדכנים את המיקום הנוכחי של העצם ביחס למקום הנוכחי ולזמן שעבר, לדוגמא, עדכון המיקום האנכי נעשה כך:

x += dx * elapsedTime

כאשר x הוא נקודת x הנוכחית, dx היא המהירות האנכית ו – elapsedTime הוא הזמן שעבר מאז העידכון הקודם.

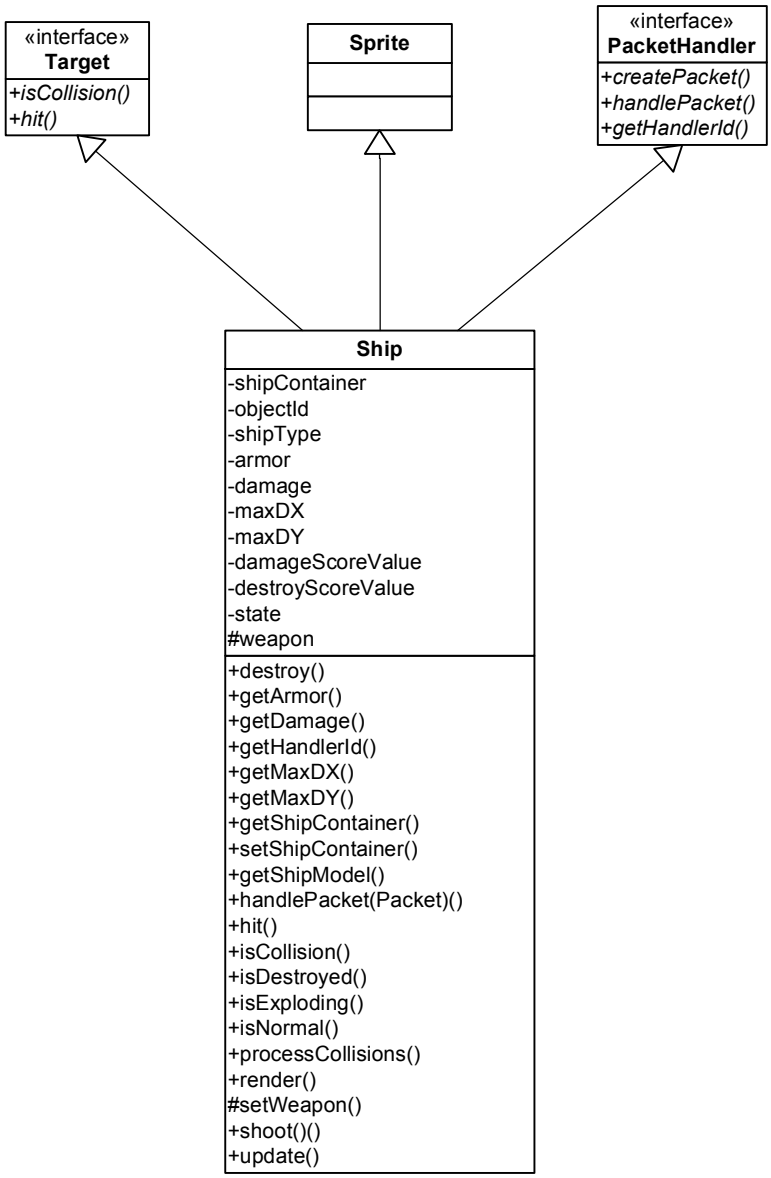
המימוש של המתודה render במחלקה **Sprite** גם הוא פשוט למדי, מכיוון שאנו משתמשים בתמונה לייצוג העצם, אנו בסה"כ משתמשים במתודה **drawImage** של **Graphics**.

כך, באמצעות קריאות למתודות **updateState** ו – **render** המגיעות מהלולאה הראשית נע העצם על גבי המסך.

נרחיב כעת על המחלקות המופשטות המרחיבות את **Sprite**: **Bullet**, **Bonus** ו – **Ship**.

Ship:

איור 5.2 מפרט את מרבית תכונות ומאפייני המחלקה המופשטת **Ship**.



איור 5.2 – המחלקה Ship

כפי שניתן לראות מחלקה זו מרחיבה את המחלקה **Sprite** וממשת את הממשקים **Target** ו – **PacketHandler**. חללית, כמחלקה המרחיבה את **Sprite**, נעה ומציירת את עצמה ע"ג המסך ובנוסף מגדירה התנהגויות נוספות הנחוצות לחללית כגון ירי, שינוי תנועה אקראי ותכונות נוספות כגון מיגון, מהירות מקסימלית ועוד.

המימוש של הממשק **Target** נועד בכדי להוסיף את החללית כמטרה של עצמים אחרים במשחק (יריות, חלליות אחרות). בממשק זה מוגדרות שתי מתודות עקריות, **isCollision**, שלה מעבירים מיקום ומחזירה אמת אם החללית מתנגשת עם העצם שנמצא במיקום זה והמתודה **hit** שבה מבצעת החללית את הפעולות הנדרשות במקרה של פגיעה.

המימוש של הממשק **PacketHandler** מאפשר לחללית לשלוח את מצבה הנוכחי לשחקן הרשת בכדי לסנכרן בין השחקנים ובכדי לטפל במנות נכנסות שנועדו לעדכן חללית זו.

המחלקות **EnemyShip** ו – **PlayerShip** יורשות מהמחלקה **Ship** ומוסיפות כל אחת התנהגויות ותכונות המתאימות לחללית של האויב ושל השחקן בהתאמה.

Bullet :

המחלקה המופשטת **Bullet** היא המחלקה הבסיסית של כל היריות הנורות מן החלליות במשחק. גם היא מרחיבה את **Sprite** ומממשת את הממשקים **Target** (כעצם שיכול לשמש כמטרה או כפוגע במטרה) ו – **PacketHandler** (כל ירייה שולחת מנה לשחקן הרשת בכדי שייצור אותה). היריות נורות ע"י אובייקט **Weapon** (אחד או יותר) שיש לכל חללית.

Bonus :

המחלקה **Bonus** דומה מאד למחלקה **Bullet** אלא שבמקום לגרום נזק לחללית הפוגעת, ניתן בונוס כלשהו (נשק טוב יותר, מיגון נוסף וכיו"ב). הבדל נוסף הוא שרק חלליות אוייב משחררות בונוס ורק חללית של השחקן יכולה לצרוך בונוס.

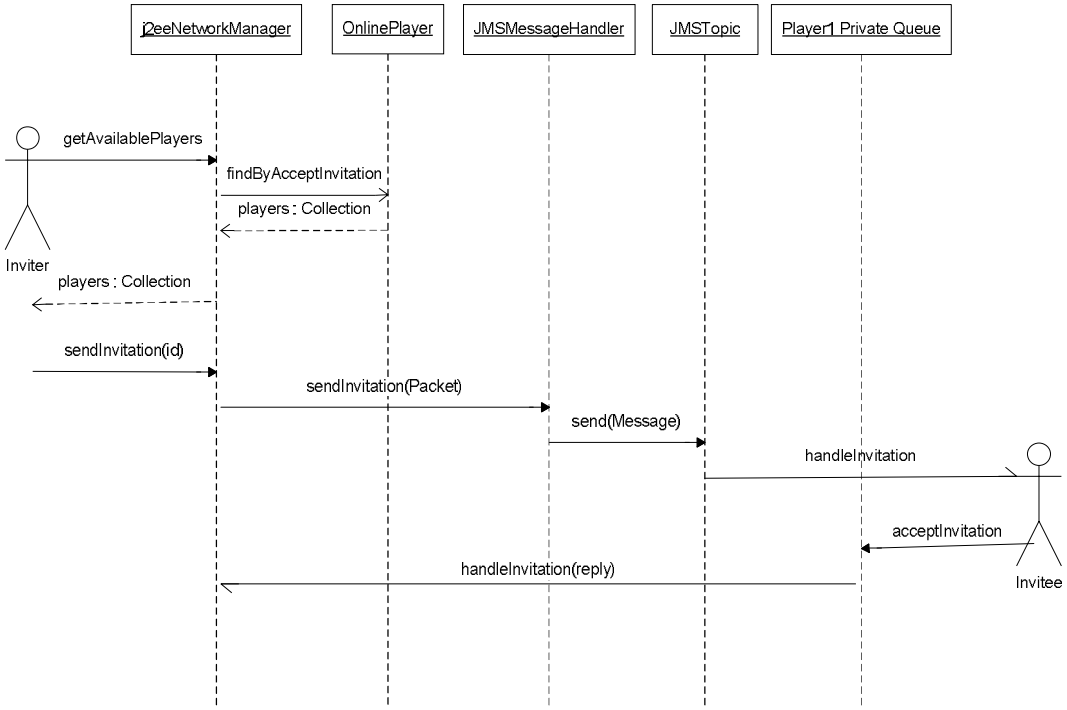
6 משחק רשת

משחק רשת מאפשר לשני שחקנים לשחק יחדיו דרך הרשת באותו משחק נגד המחשב. כל הפניות לשרת ולמחשב של שחקן הרשת נעשות דרך שני ממשקים, **NetworkManager** ו – **GameNetworkManager**. הראשון משמש לתקשורת עם השרת עוד לפני שהחל המשחק או בסיומו לצורך התחברות, הרשמה, שליחת זימון לאחד השחקנים הפנויים, שליחת התוצאה לשרת ועוד. השני (**GameNetworkManager**) משמש לתקשורת בין שני השחקנים במהלך המשחק והוא מספק שירותים כגון שליחת וקבלת מנות משחק (**Packet**) אגירת מנות נכנסות ועוד.

בפרוייקט זה המחלקה **J2EENetworkManager** מממשת את הממשק **NetworkManager** ו – **J2EEGameNetworkManager** מממשת את הממשק **GameNetworkManager**. כפי שמרמז שמן של המחלקות, אנו משתמשים בשירותים שנותן לנו שרת J2EE בכדי לממש את ממשקי הרשת.

תקשורת מחוץ למשחק

את התקשורת לפני שהחל המשחק מנהלת המחלקה **J2EENetworkManager**. מחלקה זו משתמשת ב – EJBs הנמצאים בשרת וב – Topic של JMS הנועד לשלוח זימונים למשחקי רשת בין שחקנים.
איור 6.1 מציג דיאגרמה הממחישה את אופן התחלת משחק רשת.



איור 6.1 – דיאגרמה להדגמת התחלה של משחק רשת

השחקן המזמין מחליט שהוא רוצה לשחק עם שחקן אחר, הוא מבקש את רשימת השחקנים הפנויים ממופע של מנהל הרשת (אובייקט מסוג **J2EENetworkManager**). מנהל הרשת פונה ל – **OnlinePlayer**, שהוא entity bean ומקבל ממנו את רשימת כל השחקנים הפנויים באמצעות קריאה למתודת החיפוש המתאימה (**findByAcceptInvitation**) המחזירה את רשימת כל השחקנים המחוברים ומוכנים לקבל זימונים לשחק. לאחר קבלת הרשימה יכול השחקן המזמין לבחור שחקן פנוי מן הרשימה ולשלוח לו זימון דרך מנהל הרשת. הזימון נעשה לפי מספר סידורי. כל שחקן המבצע כניסה לשרת באמצעות משתמש וסיסמא, מקבל מספר סידורי שישמש אותו לאורך השיחה (session id). מנהל הרשת יוצר מנה עם פרטי ההזמנה ושולח אותה דרך מחלקת עזר המטפלת בתקשורת עם שרת ה – JMS (**JMSMessageHandler**). אנו משתמשים ב – JMS בכדי לשלוח הודעות משחקן לשחקן.

בכדי להזמין שחקנים לשחק אנו משתמשים ב – JMS Topic. כל שחקן המתחבר לשרת ומקבל מספר סידורי מאזין ל – Topic זה ומוסיף מסנן (Selector) כך שיתקבלו רק הודעות המיועדות אליו. בנוסף לכל שחקן מחובר יש תור (Queue) JMS זמני פרטי שאליו הוא יקבל הודעות בזמן משחק רשת עם שחקן אחר.

אז בכדי שההודעה תגיע ליעדה, ה – **JMSMessageHandler** מוסיף להודעה תכונה המציינת את המספר הסידורי של היעד ומוסיף גם את הכתובת של התור הפרטי של השחקן המזמין שאליו ישלחו ההודעות בזמן המשחק. לאחר פעולות אלו נשלחת ההודעה ל – Topic ומגיעה בסופו של דבר לנמען. אצל הנמען תצוץ תיבת שיח עם פרטי ההזמנה כשזו תגיע אליו. הנמען

יכול לסרב או להסכים. התשובה שהוא בוחר נשלחת בצורה דומה למזמין והוא פועל בהתאם. במידה והוא מקבל את ההזמנה הוא הכתובת של התור הפרטי נשלחת גם היא יחד עם התשובה.

אם ההזמנה נדחית שני השחקנים המעורבים ממשיכים כרגיל בלי שהתחיל משחק ביניהם. אם היא מתקבלת אזי אצל הנמען יחל המשחק והוא ימתין במצב של **LoadingLevelState** עד אשר יישלחו הפרטים הדרושים מן המזמין אל התור הפרטי (חלליות בשלב הנוכחי ומיקומיהן). השחקן המזמין יקבל את התשובה (החיובית), יאשר אותה והמשחק יתחיל גם אצלו כאשר הוא נחשב כבקר (controller) במשחק הרשת, כלומר מהמחשב שלו נקבעים הארועים האקראיים והוא זה שטוען את החלליות מן הקובץ ושולח את פרטיהן לשחקן השני בזמן טעינת השלבים השונים במשחק.

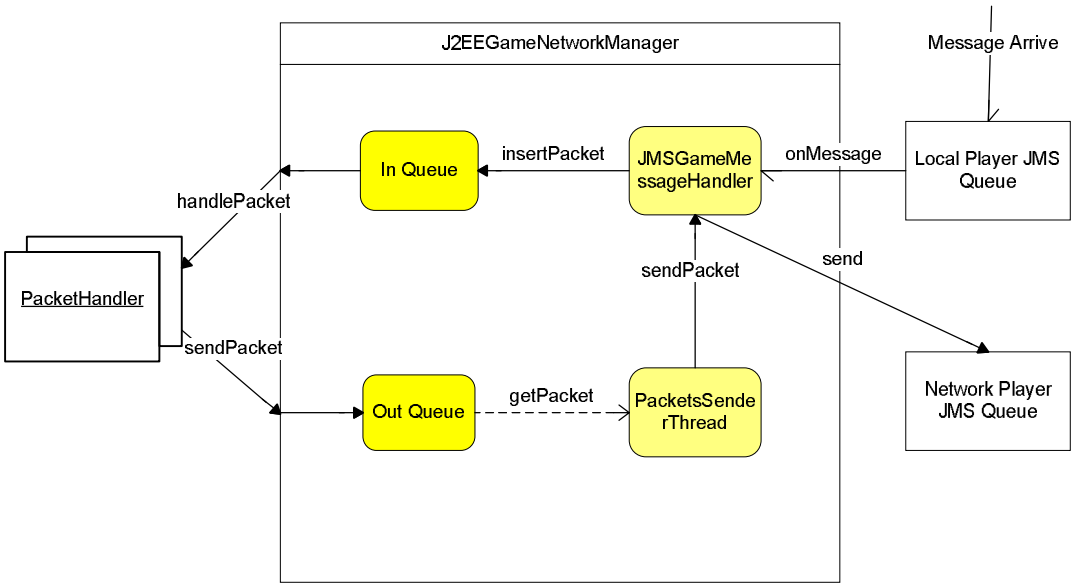
תקשורת בזמן המשחק

בזמן המשחק (לאחר ששני שחקנים הסכימו לשחק והמשחק החל) התקשורת ביניהם מתבצעת דרך הממשק **GameNetworkManager**, שהמימוש שלו כרגע הוא במחלקה **J2EEGameNetworkManager**. לכל שחקן יש תור JMS פרטי משלו ואת כתובת התור הפרטי של השחקן השני. כל ההודעות במהלך המשחק נשלחות אל התור הפרטי של השחקן השני באמצעות ה – Message Sender.

האובייקטים הנשלחים הם כולם נגזרים מהמחלקה המופשטת **Packet**. מחלקה זו מממשת את הממשק **Serializable** וכוללת מספר תכונות בסיסיות המשותפות לכל המנות הנשלחות במשחק כגון מספר השולח, מספר המקבל, מספר האובייקט שאמור לטפל במנה ודגל האומר האם מנה זו נצרכה ע"י אחד האובייקטים.

מנות לדוגמא הן **NewLevelPacket** המכילה פרטים הנחוצים לתחילת שלב חדש, **PlayerQuitPacket** המודיעה על עזיבת שחקן הרשת את המשחק, **BulletPacket** עם פרטי יירייה חדשה שנורתה.

איור 6.2 מציג תיאור סכמטי של שליחת וקבלת הודעות.



איור 6.2 – שליחת וקבלת הודעות במהלך המשחק

שליחה של מנות (**Packets**) וטיפול במנות נכנסות מתבצעת האובייקטים במשחק המממשים את הממשק **PacketHandler**. כאשר אובייקט מחליט שיש לשלוח מנה חדשה (במהלך מתודת **update** שלו), הוא יוצר אותה וקורא למתודה **sendPacket(Packet)** של מנהל הרשת במשחק בכדי לשלוח אותה. מנהל הרשת אינו שולח מיד את המנה, בכדי לא לעכב את ריצת המשחק, אלא מכניס אותה לתור היוצא ומעיר את התהליכון ששולח את המנות (תהליכון זה ממתין על התור היוצא). כאשר התהליכון האחראי לשליחת ההודעות (**PacketsSenderThread**) מתעורר הוא עובר על כל המנות בתור היוצא ושולח אותן באמצעות האובייקט האחראי לשליחת וקבלת הודעות משרת ה – JMS. האובייקט ששולח הודעות לתור ה – JMS ומאזין להודעות נכנסות הוא מהמחלקה **JMSGameMessageHandler**. כאשר הוא מתבקש לשלוח מנה, הוא מכניס אותה ל – **ObjectMessage** (טיפוס הודעה של JMS המכיל אובייקטים) ושולח אותה לתור JMS הפרטי של שחקן היעד.

כאשר מתקבלת הודעה היא מגיעה לתור JMS הפרטי של השחקן והמנגנון של JMS מעביר אותה לאובייקט שמאזין לתור זה (במשחק האובייקט מטיפוס **JMSGameMessageHangler**) באמצעות קריאה למתודה **onMessage(Message)** שלו.

המאזין מוציא את האובייקט מסוג **Packet** המוכל בהודעה ומכניס אותו לתור הנכנס (באמצעות מנהל הרשת). גם במקרה זה המנות אינן מטופלות מיד, אלא בשלב איסוף הקלט של המצב (**GameState**) הנוכחי. בשלב איסוף הקלט המצב הנוכחי עובר על כל ההודעות ומחפש את האובייקט המתאים לטיפול בכל מנה (בד"כ באמצעות מספר המטפל שמופיע במנה ולעיתים לפי סוג המנה). המנה אז עוברת לאובייקט לטיפול וכאשר הוא מסיים את עבודתו המנה מסומנת כמנה שהשתמשו בה (consumed) והיא יורדת מן התור הנכנס.

7 צד השרת

צד השרת במשחק מיושם על גבי שרת J2EE (Sun Application Server 8) המספק לנו את שרת ה – JMS ואת ה – EJBs הנחוצים והגישה לבסיס הנתונים. הוראות ההתקנה יפורטו בפרק המיועד לכך.

נפרט בקצרה את השירותים שמספק השרת.

JMS

כפי שהוזכר קודם לכן, התקשורת במשחק מתבצעת ברובה דרך שירותי שרת ה – JMS (השאר מתבצע דרך ה – EJBs).

שרת ה – JMS מספק לנו את שירות העברת ההודעות. בכדי שנוכל לפתוח קישור, לייצר הודעות ולשמור הודעות יש צורך ב – connection factory וב – destination אליו ישלחו וממנו יתקבלו הודעות (ה – Topic).

בשרת ה – JMS מוגדר connection factory אחד ליצירת הקישור ו – destination אחד לשליחת זימונים. את התור הזמני של כל אחד מהשחקנים יוצרים מהקישור הקיים.

הלקוח מחפש את ה – **ConnectionFactory** באמצעות JNDI, לאחר שמצא אותו הוא יוצר **Connection** לשרת וממנו את ה – **MessageProducer** וה – **MessageConsumer** שיעדם הוא ה – **Destination** שהוגדר בשרת (ואותו מוצאים באמצעות JNDI).

EJBs

מספר EJBs נמצאים על השרת ותומכים במשחק. נפרט כל אחד מהם בחלק זה.

:Player

Player הוא Entity Bean המייצג משתמש רשום במערכת. באמצעותו ניתן לחפש משתמש רשום לפי שם משתמש וסיסמא, לעדכן פרטי משתמש ולהסיר משתמשים קיימים.

:OnlinePlayer

OnlinePlayer גם הוא Entity Bean ומייצג משתמש המחובר כרגע לשרת. הוא מחזיק את המספר הסידורי של החיבור, שם המשתמש וזמן ההתחברות. בנוסף מוחזק דגל המציין האם

המשתמש מוכן לקבל הזמנות למשחק או לא (כאשר הוא באמצע משחק לא ניתן לקבל הזמנות). באמצעותו ניתן לקבל את רשימת כל השחקנים החופשיים למשחק רשת.

:SequenceFactory

Entity Bean הנותן שירות של יצירת מספרים סידוריים ייחודיים לפי מפתח (המפתח מסוג מחרוזת). נשתמש בו ליצירת מספרים סידוריים לטבלאות השונות, לדוגמא מספרים סידוריים ל – OnlinePlayer.

:HighScore

HighScore הוא Session Bean המאפשר לשלוח ולשמור תוצאות של שחקנים, להחזיר את עשר התוצאות הטובות ביותר או טווח של תוצאות טובות ביותר.

:SignIn

SessionBean המשמש לכניסה לשרת (מציאת ה – Player לפי שם המשתמש והסיסמא ואם נמצא יצירת OnlinePlayer חדש), יציאה מהשרת (logout הסרת ה – OnlinePlayer) ויצירת חשבון חדש (יצירת Player חדש).

בסיס נתונים

בסיס הנתונים למשחק הוא די פשוט ומכיל 4 טבלאות. הגישה לנתונים נעשית דרך ה – EJBs בלבד. הטבלאות בשימוש הן :

- player – לשמירת נתוני המשתמשים הרשומים. הטבלה מכילה 3 עמודות, שם המשתמש (מפתח ראשי), סיסמא וכתובת דואר אלקטרוני (רשות).
- online_user – שמירת נתוני המשתמשים המחוברים כעת לשרת. מכילה מספר סידורי (מפתח ראשי), שם משתמש (מפתח משני), זמן ההתחברות והאם המשתמש מוכן לקבל הזמנות לשחק.
- sequence_factory – טבלה זו משמשת את ה - SequenceFactory Bean בכדי שיוכל לשמור את המספר הסידורי הבא לכל מפתח. הטבלה מכילה שם טבלה כמפתח ראשי ואת המספר הסידורי הבא.
- high_score – טבלה זו שומרת את התוצאות שנשלחות ע״י המשתמשים. בטבלה נשמרים מס׳ סידורי (מפתח ראשי), שם השחקן, ניקוד השחקן, והשלב אליו הגיע השחקן.

8 הוראות התקנה והרצה

בפרק זה נסקור מהן דרישות התוכנה של המשחק ומהם השלבים הנחוצים להתקנה והרצה של המשחק.

דרישות המשחק

- גירסא 1.4 ומעלה של Java (Java 2 JRE 1.4+).
- גירסא 1.4 ומעלה של J2EE. בכדי לשחק במשחק רשת יש צורך בחבילות של J2EE 1.4. כל החבילות הנחוצות נמצאות בקובץ j2ee.jar.
- על המחשב שישמש כשרת יש להתקין שרת J2EE העונה לתקן 1.4 או מאוחר יותר.

התקנת השרת

התקנת השרת כוללת מספר שלבים, ביניהם הגדרת שירות ה – JMS ובסיס הנתונים. ההוראות שלהלן מתייחסות לשרת J2EE של סאן, Sun Application Server 8, הניתן להורדה חינם מ - http://java.sun.com/j2ee/1.4/download.html#sdk. בהמשך ההוראות נניח שהותקן השרת עם כל רכיביו השונים לפי הוראות ההתקנה שלו.

לאחר ההתקנה יש להריץ את השרת (Start Default Server) ואת בסיס הנתונים (Start PointBase) לביצוע ההוראות שלהלן. (בעת ההתקנה מותקנים כל קיצורי הדרך בתיקייה אחת, בחלונות הקיצורים יימצאו כבררת מחדל ב– start->programs->J2EE 1.4 SDK).

- **הגדרת שרת ה – JMS**

שרת JMS כבר הוגדר מראש. עלינו להגדיר לו Connection Factory ו - Destination

  - פתח את ה – Admin Console כאשר השרת רץ
  - הרחב את הענף Java Message Service
  - בחר ב – Connection Factories
  - בחלק הימני לחץ על New
  - בשם JNDI הכנס jms/TopicConnectionFactory, ב – Type בחר ב - javax.jms.TopicConnectionFactory. סמן את התיבה Enabled. לחץ על OK.
  - בחלק השמאלי בחר ב – Destination Resources
  - במסך הימני לחץ על New

- ב- JNDI Name הכנס jms/invitationTopic, ב- Type בחר javax.jms.Topic, סמן את התיבה Enabled, לחץ על Add Property, בשדה Name הכנס את המחרוזת Name ובשדה Value הכנס PhysicalTopic. לחץ על OK.

- **הגדרות JDBC**

ה – EJBs בפרוייקט פונים לשרת נתונים בשם jdbc/gameDB. עלינו להגדיר שם זה בשרת.

- פתח את ה – Admin Console
- הרחב את הענף JDBC ובחר ב – JDBC Resources
- לחץ על New
- בחלק הימני הכנס jdbc/gameDB ב- JNDI Name, בחר ב – PointBasePool מרשימת ה – Pool Name (Pool הנוצר כברירת מחדל) וודא שהתיבה Enabled מסומנת.

- **יצירת הטבלאות ב- PointBase**

בשלב זה ניצור את הטבלאות הנחוצות למשחק. לצורך כך יש לפתוח את ה- console של PointBase, ולהעתיק את פקודות ה - SQL שמובאות להלן.

- נווט לתיקייה serveroption שבתיקיית ההתקנה של PointBase הנמצאת בתוך תיקיית ההתקנה של Sun Application Server. לדוגמא:
  C:\Sun\AppServer\pointbase\tools\serveroption
- לחץ על קובץ האצווה startconsole.bat
- הכנס את הפרטים הבאים:

  Driver: com.pointbase.jdbc.jdbcUniversalDriver
  URL: jdbc:pointbase:sun-appserv-samples
  User: PBPUBLIC
  Password: PBPUBLIC

  בחר בכפתור Open specified database ולחץ על OK.
- העתק והדבק את הפקודות הבאות לתיבת הטקסט Enter SQL Commands:
  (ניתן גם לטעון את הקובץ game_tables.sql הנמצא על הדיסק)

```
CREATE TABLE player
(user_name VARCHAR ( 50)  NOT NULL
, password VARCHAR ( 50)  NOT NULL
, email VARCHAR ( 100)
,CONSTRAINT PK PRIMARY KEY ( user_name)
 );


CREATE TABLE online_player
(session_id BIGINT  NOT NULL
, user_name VARCHAR ( 50)  NOT NULL
, accept_invitations BOOLEAN  NOT NULL
, session_start_time BIGINT  NOT NULL
,CONSTRAINT pk PRIMARY KEY ( session_id)
,CONSTRAINT fk FOREIGN KEY ( user_name) REFERENCES
player(user_name)
 );


CREATE TABLE sequence_factory
(table_name VARCHAR ( 20)  NOT NULL
, next_id BIGINT  NOT NULL
,CONSTRAINT pk PRIMARY KEY ( table_name)
 );


CREATE TABLE high_score
(score_id BIGINT  NOT NULL
, player_name VARCHAR ( 50)  NOT NULL
, score BIGINT  NOT NULL
, level INTEGER  NOT NULL
,CONSTRAINT pk PRIMARY KEY ( score_id)
 );
```

- לחץ על Execute All

- **פריסת האפליקציה על השרת (deployment)**
  - פתח את הכלי Deploytool (קיצור דרך אליו נוצר עם ההתקנה)
  - בחר בתפריט File, ובחר ב - Open
  - לחץ על הכפתור Browse ונווט אל התיקייה בה נמצא הקובץ Game.ear (הקובץ נמצא בתיקייה הראשית ע"ג דיסק הפרוייקט, יש להעתיקו לתיקייה בכונן המקומי ולטעון אותו משם).
  - מצד שמאל הרחב את הענף Files ואז את Applications ובחר ב – Game
  - לפריסת האפליקציה על השרת בחר Tools ו – Deploy
  - בחלון שנפתח וודא ששרת היעד הוא השרת הרצוי (ברירת המחדל היא localhost:4848) וסמן את התיבה Return Client Jar ובחר נתיב לקובץ. בקובץ שיוחזר תהיה אפליקצית הלקוח אותה נריץ בכדי לשחק במשחק רשת. לחץ על OK.

- **הגדרות נוספות**
  - בכדי להשתמש במצב מסך מלא תחת ה – Application Client Container (ACC) המשמש כסביבת ריצה לאפליקציה חיצונית הניגשת לשרת J2EE יש להוסיף הרשאות בקובץ ה – policy. הקובץ נמצא בתיקייה lib\appclient שבתיקיית ההתקנה של Sun Application Server. פתח את הקובץ client.policy בעזרת עורך טקסט והוסף את השורה הבאה בתוך הסוגריים המסולסלים של grant הכללי (התחתון ביותר):

```
permission java.awt.AWTPermission "fullScreenExclusive";
```

הרצת המשחק

- להרצת המשחק לשחקן יחיד העתק מן הדיסק את התיקייה המכילה את התיקייה הראשית של קובצי ה – class ואת התיקיות resources (תיקייה זו מכילה את התמונות וקובצי הקול) ו – config (מכילה את קובץ ה – XML של השלבים ואת ה – DTD שלו) מן הדיסק לכונן המקומי. הרץ את הקובץ runGame.cmd.

- להרצת משחק רשת (וגם יחיד) יש לבצע את הפעולות שפורטו בסעיף התקנת השרת ולהעלות את השרת ואת בסיס הנתונים. העתק את תוכן התיקייה net_game_files מן הדיסק לתיקייה בה נמצא הקובץ GameClient.jar שנוצר ע"י הכלי deployTool והרץ את הפקודה runGame.cmd.

- ניתן להריץ את המשחק במצב debug. (כרגע השימוש העיקרי של מצב זה הוא לבדיקת משחק רשת על אותו המחשב. במצב debug ייפתח חלון מוקטן הניתן להזזה). להרצת המשחק במצב debug הרץ את הפקודה runDebug.cmd.

ביבליוגרפיה

ספרות כללית על Java ו - J2EE:

- Bond, M., et al, Sams Teach Yourself J2EE in 21 Days, Sams, 2002
- Deitel H.M, Deitel P.J, Santry S.E, Advanced Java 2 Platform How to Program, Prentice-Hall, 2002
- Deitel H.M, Deitel P.J. Java How to Program, Fourth Edition, Prentice-Hall, 2002
- Eric A., et al, The J2EE 1.4 Tutorial, Sun Microsystems, 2004
- Ragae G., Krishna K., Sams Teach Yourself EJB in 21 Days, Sams Publishing, 2002
- Roman E., et al, Mastering Enterprise JavaBeans, Second Edition, Wiley Computer Publishing, 2002

ספרות בתחום משחקים ב – Java:

- Brackeen D., Barker B., Vanhelsuwé L., Developing Games in Java, New Riders Publishing, 2003
- Fan Joel, Black Art of Java Game Programming, Macmillan Computer Publishing, 1996
- Morrison M., Teach Yourself Internet Game Programming with Java in 21 Days, Sams, 1996
- Mulholland A, Murphy G., Java 1.4 Game Programming, Wordware Publishing, 2003

מקורות מ – SUN:

- Martak M., "Full Screen Exclusive Mode API", http://java.sun.com/docs/books/tutorial/extra/fullscreen/
- Sun Microsystems, "JavaSound API Programmer's Guide", http://java.sun.com/j2se/1.4.2/docs/guide/sound/programmer_guide/index.html, 2001
- Haase J., "Java Message Service API Tutorial", 2002

מאמרים שונים מהאינטרנט:

- Book On Hook, "Introduction to Multiplayer Game Programming", http://www.bookofhook.com/Article/GameDevelopment/MultiplayerProgramming.html, 2003
- Jenkins K., " Designing Secure, Flexible, and High Performance Game Network Architectures", http://www.gamasutra.com/features/20041206/jenkins_01.shtml, 2004
- Haase C., Trembovetski D., "Java Gaming - Understanding the Basic Concepts", JDJ Magazine, Vol 9, Issue 9, 2004, http://java.sys-con.com/read/46663.htm
- Haase C., Trembovetski D., "Java Gaming – 2D Rendering", JDJ Magazine, Vol 9, Issue 10, 2004, http://java.sys-con.com/read/46663.htm
- Tony and Piage.com, "Managing Game States in C++", http://tonyandpaige.com/tutorials/game1.html
- Java Corporation, "The Java Game Development Tutorial", http://javacooperation.gmxhome.de/TutorialStartEng.html
- Glass K., "Space Invaders 101", http://grexengine.com/sections/externalgames/articles/Kevin%20Glass-Space%20Invaders%20101-1.html, 2004