**שלב 5 - תמיכה בצבע\אור, הוספת סכמה, ובניית תמונה עם תאורת מילוי**

לפני שממשכים לשלב הזה יש לוודא שלא שכחתם ליצור **tag** ב-**commit** האחרון עבור השלב הקודם.

***נ.ב.1*** *אין לבנות בנאי ברירת מחדל, בנאי העתקה, והעמסת* ***equals*** *במחלקות של השלב הזה.*

***נ.ב.2*** *חובה להקפיד על תיעוד בפורמט* ***javadoc*** *לכל המחלקות והפעולות החדשות ולעבוד ע"פ הסדר כפי שנלמד בקורס!*

**שלב 1 – שילוב מחלקות צבע ויצירת תמונה**

קודם כל ניקח מתיבת ההגשה שני מודולים עם המחלקות המתאימות:

* המודול **Color.java** מכיל מחלקה **Color** שעוטפת (**Wrapper design pattern**) את המחמלקה של ג'אוה פריימוורק **java.awt.Color** ומספקת עבורנו את הממשק הדרוש לנו במיני-פרויקט. את המודול הזה נוסיף בחבילת **primitives**.
* המודול **ImageWriter.java** מכיל מחלקה שאחראית על שלושה תחומים:
  + הגדרת תיקיה ושם הקובץ של התמונה ופעולה **writeToImage()** ליצירת הקובץ בסיום העבודה
  + צבירת הצבעים של הפיקסלים במטריצת הצבעים של פיקסלים בעזרת הפעולה **writePixel(int j, int i, Color color)**. שני הפרמטרים הראשונים – אינדקס של הפיקסל – הראשון על ציר ה-**x** והשני על ציר ה-**y**. הפרמטר השלישי -צבע הפיקסל מהסוג שלצבע של ג'אווה פריימוורק.
  + החזקת הפרמטרים של **View Plane** הקשורים ליצירת התמונה –הרזולוציה.
  + עבור המודול הזה נוסיף חבילה חדשה: **renderer** ונוסיף את המודול **ImageWriter.java** בחבילה החדשה הזו
* A close up of a logo

  Description automatically generatedבחבילת **unittests** נוסיף מודול בדיקה **ImageWriterTest** שיכיל בניית תמונה ראשונית – תמונה בצבע אחד עם רשת קוים בצבע שני. בטסט הזה נבנה רשת של **10x16** ריבועים במסך (**ViewPlane**) רזולוציה של **500** על **800**. הצבע של הרקע ושל הרשת – לבחירתכם\ן.
* נוסיף מתוך סייר הפרויקט בסביבת העבודה נוסיף בפרויקט תת-תיקיה בשם **images**.
* נריץ את הטסט שיצרתם. נראה בתיקיה החדשה הנ"ל את קובץ התמונה בפורמט **png** שיכיל תמונה כזו:
* דרך תפריט הקשר (כפתור ימין של עבר כשהסמן מעל שם התיקיה) ניכנס לתת-תפריט של **Team** ונבחר באפשרות **ignore**. כתוצאה, בקובץ **.gitignore** שבתיקיית הפרויקט תתווסף שורה עם הטקסט הבא: **/images/**

**נ.ב.** שני השותפים חייבים ליצור את התיקיה בפרויקט במחשביהם

* אם לא רואים את קובץ התמונה (אבל הוא נמצא בדיסק): מי שעובד באקליפס, באזור של סייר חבילות ו-\או סייר פרויקט (**Package / Project Explorer**) לוחצים על שלוש נקודות בצד ימין למעלה ובוחרים **Filters** או שורה דומה מהתפריט, מורידים את כל ה-"**v**" (ישירות אחד-אחד או בעזרת כפתור **Deselect All**) ולוחצים על כפתור **OK**.

**שלב 2 – יצירת סצינה**

בחבילה **elements** נוסיף מחלקה **AmbientLight** עבור "תאורה סביבתית". הבנאי של המחלקה יקבל שני פרמטרים:

* פרמטר מסוג **Color** – לאור מילוי מקורי (עוצמת האור לפי קומפוננטות **RGB**) –
* מסוג **double** – מקדם הנחתה של אור מילוי –

המחלקה:

* תכיל שדה עצמת תאורת מילוי **intensity** מסוג **Color**,
* בנאי שמקבל ערכים כנ"ל ומחשב את העצמה הסופית של תאורת מילוי שנשמרת בשדה הנ"ל
* בנאי ברירת מחדל המאתחל את השדה הנ"ל לצבע שחור **Color.BLACK**
* ופונקציה **getIntensity()** שתחזיר את ערך עצמת התאורה הסביבתית מסוג **Color**.

נוסיף חבילה חדשה **scene** ובתוכה מחלקה חדשה **Scene** שתכיל את השדות הפרטיים הבאים:

* שם הסצנה (**String name**)
* צבע רקע (**Color background**), ברירת מחדל – צבע שחור
* תאורה סביבתית (**AmbientLight ambientLight**), ברירת מחדל – תאורה בעצמה 0 בצבע שחור
* מודל תלת-הממד (**Geometries geometries**) – בברירת מחדל יאותחל למודל ריק

המחלקה הזו הינה **PDS** לכן כל השדות בהרשאה ציבורית, אין להגדיר מתודות מאחזרות (גטרים)

במחלקה נממש את הפונקציות הבאות:

* בנאי המקבל את שם הסצנה (בלבד) שיבנה גם אוסף ריק של גופים עבור מודל **D3**.
* מתודות עדכון (סטרים) לצבע רקע, לתאורת מילוי\סביבתית, ולמודל תלת-ממד יינתנו בפורמט כמו בתבנית Builder – יחזירו את האובייקט של סצנה עצמו

**שלב 3 – עדכון פרימיטיבים**

נחזור ונוודא שפעולת המרחק בין נקודות (במחלקה **Point3D**) נבדקה ותקינה.

במחלקת קרן (**Ray**) נוסיף מתודה חדשה **findClosestPoint** המקבלת אוסף נקודות (**List<Point3D>**) ומחזירה את הנקודה הקרובה לתחילת הקרן.

לאחר הוספת שלד המתודה ותיעוד, נוסיף מחלקת בדיקה עבור הקרן שתכיל בדיקת הפעולה הזו. הפעולה תכלול 4 בדיקות:

* מחלקת שקילות – נקודה באמצע הרשימה היא הקרובה לתחילת הקרן
* 3 מקרי גבול\קצה:
  + רשימה ריקה (המתודה צריכה להחזיר ערך **null**)
  + הנקודה הראשונה היא הכי קרובה לתחילת הקרן
  + הנקודה האחרונה היא הכי קרובה לתחילת הקרן

**שלב 4 – בניית מנוע רנדור**

1. ניקח מתיבת ההגשה מודול **RenderTests.java** ונוסיף אותו בחבילה **unittests**. פונקציית הבדיקה שבמודול תבנה מודל **D3** עם ארבעה משולשים וכדור, מצלמה, סצנה, אובייקט יצירת תמונה ואובייקט רינדור ותפעיל את בניית התמונה כפי שלמדנו.

הפעם הבדיקה לא תודיע לנו האם היו תקלות או לאו אלא לאחר הרצת הבדיקה יש לעיין בתמונה שנוצרה בתיקיית הפרויקט ולוודא את תקינותה! כמו כן – שימו לב שהמודול בתיבת ההגשה משווך לתת-חבילה elements בחבילת הבדיקות – אל תשכחו לעדכן את שם החבילה במודול אם הארכיטקטורה שלכם שונה במקצת.

1. בחבילה **renderer** נוסיף מחלקה אבסטרקטית **RayTracerBase** שתכיל שדה של סצינה בהרשאה **protected**, בנאי המקבל בפרמטר אובייקט של סצינה ומתודה אבסטרקטית ציבורית **traceRay** המקבלת קרן בפרמטר ומחזירה צבע (**Color**). יש להוסיף תיעוד למחלקה לבנאי ולמתודה.
2. בחבילה **renderer** נוסיף מחלקה **RayTracerBasic** היורשת מהמחלקה האבסטרקטית הנ"ל. בשלב הזה במחלקה יהיו בנאי המקבל סצינה ומפעיל את הבנאי של מחלקת האב ומימוש ריק (עם חריגה או עם החזרת ערך **null**). יש להוסיף תיעוד למחלקה ולבנאי.
3. בחבילה **renderer** נוסיף מחלקה **Render** (שתפקידה ליצור מהסצנה את מטריצת הצבעים של התמונה). המחלקה תכיל שדות של **ImageWriter**, של סצנה, של קמרה ושל סורק קרניים הנ"ל (מטיפוס של המחלקה האבסטרקטית הנ"ל). המחלקה תיבנה חלקית לפי התבנית **Builder** ותכלול את המתודות הציבוריות הבאות:

* נ.ב. למחלקה לא יוגדר בנאי (בנאי בררת מחדל מרומז ישמש אותנו)
* מתודות מאחזרות מעדכנות (סטרים) המחזירות את אובייקט עצמו לצורך שרשור
* מתודה **renderImage()** שלא מחזירה שום ערך שבשלב הזה תבדוק תחילה שבכל השדות הוכנס ערך לא ריק (ובמקרה של חוסר תזרוק חריגה מתאימה, למשל **MissingResourcesException**) ולאחר מכן תזרוק חריגה **NotImplementedUnsupportedOperationException**
* A close up of a sign

  Description automatically generatedמתודה **printGrid(int interval, Color color)** שתיצור רשת קוים בדומה למה שבוצע בבדיקה בשלב הראשון (שימו לב לא לדרוס את התמונה – יש לתת צבע רק לקוי הרשת אך לא ליתר הפיקסלים). כמובן בתחילה המתודה תבדוק תחילה שבשדה של יצרן התמונה הוכנס ערך לא ריק (ובמקרה של חוסר תזרוק חריגה מתאימה, למשל **MissingResourcesException**)
* מתודה **writeToImage()** שבתחילה המתודה תבדוק תחילה שבשדה של יצרן התמונה הוכנס ערך לא ריק (ובמקרה של חוסר תזרוק חריגה מתאימה, למשל **MissingResourcesException**) ולאחר מכן תפעיל (האצלה!) את המתודה המתאימה של יצרן התמונה.

לאחר שלמחלקה ולכל המתודות נוסף תיעוד, וכמובן הטסט ששמנו – לא עובד:

* נממש תחילה את המתודה **renderImage** שתבצע לולאה על כל הפיקסלים של ה-**ViewPlane**, עבור כל פיקסל תיבנה קרן ועבור כל קרן נקבל צבע מסורק הקרניים. את הצבע נשים בפיקסל מתאים של יצרן התמונה (**writePixel**)
* נממש את המתודה של סריקת קרן במחלקה **RayTracerBasic** שתחפש חיתוכים בין הקרן לבין מודל תלת-הממד של הסצנה. אם לא נמצאו נקודות חיתוך – תוחזר צבע הרקע של הסצנה. אחרת תימצא הנקודה הקרובה לתחילה הקרן בעזרת המתודה החדשה שיצרנו בשלב 3 הנ"ל. לאחר מכן נמצא את הצבע בנקודה הזו בעזרת מתודה פרטית חדשה כדלקמן:
  + במחלקה **RayTracerBasic** נוסיף מתודה פרטית חדשה **calcColor** שמקבלת בפרמטר נקודה ומחזירה צבע. כמובן יש להוסיף תיעוד למתודה. בשלב הזה של המיניפרויקט המתודה תחזיר את צבע תאורת מילוי\סביבתית של הסצנה ותו לא (אין בינתיים שימוש בנקודה שקיבלנו בפרמטר)

**שימו לב**: במחלקת הטסטים של יצירת תמונה בסיסית יש גם בדיקה לתמונה הנבנית עם סצנה שנוצרת בעזרת קובץ XML (ראו בבונוס) – אם אינכם מבצעים את הבונוס – עליכם למחוק את המתודה. אם אתם כן מבצעים את הבונוס – תצטרכו להוסיף קוד במתודה הזו בהתאם.

**שלב לבונוס – בניית סצנה מקובץ XML**

ניתן לקבל בונוס של עד 2 נק' ע"י:

* פעולת עזר לטסט מנוע רינדור המקבל שם קובץ **XML** ובונה ממנו את האובייקטים של סצנה ו-**imageWriter**
* חבילת ניתוח קובץ **XML** ובניית האובייקט של סצנה אחת הספריות של **java**: **DOM**, **SAX** או **JAXP**.

נ.ב. בתיבת התרגיל ניתן קובץ **basicRenderTestTwoColors.xml** עבור הבונוס.

**בהצלחה!**