משימות לשלב 1

חלק א': הגדרת פרימיטיבים

בשלב ראשון של הפרויקט נרצה להגדיר אוסף של מחלקות בעזרתם נוכל לתאר סצנה גרפית.

נתחיל בהגדרת חבילה של פרימיטיבים בשם primitives שכוללת את המחלקות הבאות:

- קואורדינטה - יחידה על ציר המספרים.
- נקודה במישור - נקודה בעלת 2 קואורדינטות.
- נקודה במרחב - נקדה בעלת 3 קואורדינטות.
- וקטור - ישר שעובר דרך ראשית הצירים ונקודה נתונה במרחב – מגדיר כיוון.
- קרן - וקטור שאינו עובר בראשית הצירים. מוגדר ע"י נקודה וכיוון (וקטור).

עלינו להחליט מה היחס בין המחלקות השונות מבחינת ירושה והכלה.

כדי לאפשר עבודה עם האוביקטים השונים נוסיף בכל מחלקה:

- constructors (עם פרמטרים/"העתקה"). האם צריך בנאיי ברירת מחדל? אם יש לך מחשבה שכן – מה לדעתך משמעות של בנאי כזה בכל אחד מהאובייקטים?
- set – באיזה שדות צריך (אם בכלל? באיזה אובייקטים? אילו שדות?)
- העמסת פונקציה equals – לא לשכוח בדיקות בהתחלה והתבסוסות של equals של אובייקטים כלולים ושל "אבא"
- שימו לב על מה שנאמר בכיתה לגבי שוויון קואורדינטות...
- העמסת פונקציה toString על מנת לאפשר הדפסה של האובייקט – בצורה נוחה לשימוש!

בנוסף לכל מחלקה נוסיף אופרטורים שמאפשרים עבודה עם אובייקטים בטיפוס זה:

קואורדינטה:

- add – חיבור ערכים. האם ניצור קואורדינטה חדש בתוצאה? או נשנה קואורדינטה קיימת? מה ההשלכות של כל אחד מההחלטות האלה? במקרה של החלטה ראשונה – האם להחזיר קואורדינטה?
- substract – חיסור ערכים - כנ"ל.

נקודה במרחב:

- חיסור וקטורי – מקבל נקודה שניה בפרמטר, מחזיר וקטור מהנקודה השניה לנקודה שעליה מתבצעת הפעולה
- הוספת וקטור לנקודה – מחזיר נקודה חדשה
- מרחק בין 2 נקודות

וקטור:

- חיבור/חיסור וקטורי (מחזירים וקטור חדש או משנה וקטור מקורי?)
- מכפלת וקטור בסקלר (מחזירה וקטור חדש או משנה וקטור מקורי?)
- מכפלה סקלארית (dot-product) – יש להשתמש בנוסחה של אלגברה לינארית ולא טריגונומטרית. יש לבדוק וקטורים עם אותו כיוון, עם זוית חדה ביניהם, אורתוגונלים, עם זוית כהה, עם כיוונים הפוכים (על אותו ישר).
- מכפלה וקטורית – מחזיר וקטור חדש שניצב לשני הוקטורים הקיימים (cross-product) – יש להשתמש בנוסחה של אלגברה לינארית ולא טריגונומטרית. בדיקות כנ"ל.
- חישוב אורך הוקטור
- נרמול הוקטור (האם ליצור וקטור חדש מנורמל או לנרמל את הווקטור הנתון? במקרה של התשובה השניה – האם להחזיר את הווקטור המנורמל?
- תיעזרו בשקפים מצגת הרצאה שניה בשביל רענון של מושגים ופעולות על וקטורים

מקרי הבדיקה (Test cases) שיבדקו בשלב 1:

Test01: Point3D compareTo

Test02: Point3D toString

Test03: Point3D add

Test04: Point3D subtract

Test05: Point3D distance

Test06: Vector Add test

Test07: Vector Substruct test

Test08: Vector Scaling test

Test09: Vector Dot product test

Test10: Vector Length test

Test11: Vector Normalize test

Test12: Vector Cross product test

הערות כלליות לגבי עיצוב התוכנה:

1. חובה להשתמש במחלקת עזר Utility המצורפת בעמוד הבא עבור בדיקת שוויון קואורדינטות, חיסור וחיבור קואורדינטות
2. חובה לזרוק חריגה במקרה של וקטור אפס.
3. אסור לזרוק חריגה Exception, יש למצוא חריגה מתאימה
4. עיצוב התוכנה - אנחנו רוצים לשמור על ארגון ותכנון נכון של המחלקות לכן נגדיר תבנית קבועה עבור כל המחלקות:

```
package primitives;

/**
  * Javadoc formatted documentation
  */
public class ClassName extends BaseClass{
   // ***************** Constructors ********************** //
   ...
   // ***************** Getters/Setters ********************** //
   ...
   // ***************** Administration  ******************** //

   /**
     * Javadoc formatted documentation
     */
   @Override
   public bool equals(Object other) {...}

   /**
     * Javadoc formatted documentation
     */
   @Override
   public String toString() {...}

   // ***************** Operations ******************** //
}
```

5. תיעוד: יש להקפיד על תיעוד חיצוני ופנימי באופן הבא:
   א. לכל משתנה כתוב תיעוד קצר ליד הגדרתו.
   ב. לכל פונקציה במחלקה כתוב תיאור כללי של הפונקציה, מהם הפרמטרים (הקלטים) ומשמעותם, סוג הערך המוחזר (הפלט) ומשמעותו. למשל:

```
/**********************************************

* FUNCTION

*    genPrime

* PARAMETERS

*    int – highest possible prime value

* RETURN VALUE

*    A (positive) integer: the highest prime number,

*    smaller than the integer received as parameter.

* MEANING

*     This functions computes a prime number p, such that

*                 0 <= p <= PARAMETER

* SEE ALSO

*     list of names of other functions in your system,

*     related to this function.

**********************************************/

int genPrime(int)
{
          ……
}
```

6. דו"ח: יש לצרף דו"ח ב WORD לכל שלב. מטרת הדו"ח היא לתאר בפני הקורא את מטרת הפרויקט והסבר מעמיק על כל רכיבי התוכנה והאלגוריתמים המרכיבים את הפרויקט. הדו"ח יכיל:
   א. תאור של מטרת המיני-פרויקט, והאמצעים לממשו.
   ב. תאור של החבילות (Packages) והמחלקות בכל חבילה. לכל מחלקה הסבר קצר על המשתנים שלה ועל הפונקציות שהיא מכילה (ניתן להעתיק מהתיעוד). יש להסביר אלגוריתמים לא טריוויאליים (כגון constructRayThroughPixel). ניתן ורצוי לצרף תמונות ותרשימים לייתר הסבר.
   ג. צילומי מסך של הצורות הגרפיות שנוצרו באמצעות המוצר + הסבר מה רואים ומדוע?
   ד. הערות והארות שנראות לכם רלוונטיות לכל שלב.

```java
package primitives;


public abstract class Util {
	// It is binary, equivalent to ~1/1,000,000,000,000 in decimal (12 digits)
	private static final int ACCURACY = -40;

	// double store format (bit level): seee eeee eeee (1.)mmmm … mmmm
	// 1 bit sign, 11 bits exponent, 53 bits (52 stored) normalized mantissa
	// the number is m+2^e where 1<=m<2
	// NB: exponent is stored "normalized" (i.e. always positive by adding 1023)
	private static int getExp(double num) {
		// 1. doubleToRawLongBits: "convert" the stored number to set of bits
		// 2. Shift all 52 bits to the right (removing mantissa)
		// 3. Zero the sign of number bit by mask 0x7FF
		// 4. "De-normalize" the exponent by subtracting 1023
		return (int)((Double.doubleToRawLongBits(num) >> 52) & 0x7FFL) - 1023;
	}

	public static double usubtract(double lhs, double rhs) {
		int lhsExp = getExp(lhs);
		int rhsExp = getExp(rhs);

		// if other is too small relatively to our coordinate
		// return the original coordinate
		if (rhsExp - lhsExp < ACCURACY) return lhs;

		// if our coordinate is too small relatively to other
		// return negative of other coordinate
		if (lhsExp - rhsExp < ACCURACY) return -rhs;

		double result = lhs - rhs;
		int resultExp = getExp(result);
		// if the result is relatively small - tell that it is zero
		return resultExp - lhsExp < ACCURACY ? 0.0 : result;
	}

	public static double uadd(double lhs, double rhs) {
		int lhsExp = getExp(lhs);
		int rhsExp = getExp(rhs);

		// if other is too small relatively to our coordinate
		// return the original coordinate
		if (rhsExp - lhsExp < ACCURACY) return lhs;

		// if our coordinate is too small relatively to other
		// return other coordinate
		if (lhsExp - rhsExp < ACCURACY) return rhs;

		double result = lhs + rhs;
		int resultExp = getExp(result);
		// if the result is relatively small - tell that it is zero
		return resultExp - lhsExp < ACCURACY ? 0.0 : result;
	}

	public static double uscale(double lhs, double factor) {
		double deltaExp = getExp(factor - 1);
		return deltaExp < ACCURACY ? lhs : lhs * factor;
	}

	public static boolean isZero(double number) {
		return getExp(number) < ACCURACY;
	}

	public static boolean isOne(double number) {
		return getExp(number - 1) < ACCURACY;
	}

	public static double alignZero(double number) {
		return getExp(number) < ACCURACY ? 0.0 : number;
	}
}
```