המכללה האקדמית להנדסה סמי שמעון

תכנות מונחה עצמים מתקדם

עבודת הגשה מס '1

תאריך הגשה: 23:55 16/04/2023

נא לקרוא את כל המסמך לפני תחילת העבודה!

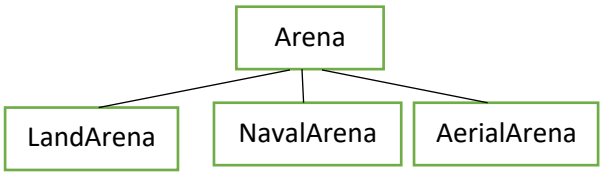
דגשים להגשה

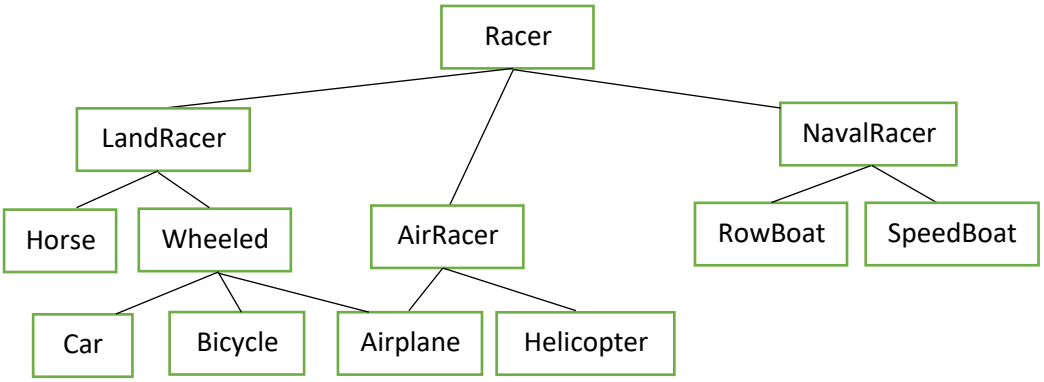
- ניתן להגיש עבודה זו בזוגות – רק אחד מהסטודנטים יגיש את העבודה במודל. בתיעוד בכל קובץ יש לציין שם ות.ז. של מגיש\ים, בתוך תיעוד ה javadoc.
- חלק מניקוד העבודה מתבצע ע"י בדיקות אוטומטיות ולכן חשוב מאוד להגדיר את כל המחלקות, המתודות והשדות בדיוק כפי שצוינו במסמך.
- חובה לתעד כל קובץ, מחלקה ופונקציה ע"י javaDoc - ניתן להיעזר בתיעוד באתר oracle או בקבצים הרלוונטיים במודל.

דגשים לעבודה זו

- על כל העבודה להיות פרויקט יחיד המחולק ל-**packages** לפי המטלות.
- על כל השדות בכל המחלקות להיות פרטיים בלבד. גישה אליהם תתבצע בעזרת **get/set**.
  גם בתוך המחלקה כל שינוי בשדה יתבצע דרך **set**, ועליה להיות פרטית במידה ואין צורך בגישה מבחוץ.
- יש להקפיד על עבודה עקבית מבחינת החלטות לגבי שיתוף מצביעים.
- על כל ה-**Setters** להיות בוליאניים, לבדוק תקינות נתונים (אם הוגדרה) ולהחזיר האם בוצעה השמה.
  במידה ונכנסים ערכים שגויים מהבנאי – יש להכניס ערך ברירת מחדל. בכל מקום אחר – יש להשאיר את הערך הקודם.
- המחלקות הנגזרות יכולות לדרוס מתודות של מחלקות בסיס לפי הצורך.
- על כל הקבועים להיות תחת שדות **final**. שימו לב, כלל ההתייחסויות במהלך הקוד לערכים ספציפיים ייעשו באמצעות גישה לקבועים הנ"ל.
- מצורף קובץ **EnumContainer.java** – בו עליכם להגדיר את כלל ה-enum בקוד לפי הדוגמא שם.
- העקרונות שנתרגל בעבודה זו הם:
  - מחלקות
  - מחלקות אבסטרקטיות
  - ממשקים
  - היררכיית הורשה – האם לייצג ישות כממשק \ מחלקה \ מחלקה אבסטרקטית
  - **Reflection** – טעינת מחלקות דינאמית, יצירת אובייקטים.
  - חריגות – טיפול בחריגות, יצירת חריגות מותאמות לתוכנית.
- <u>עליכם לבנות היררכיית מחלקות בצורה הטובה ביותר, כלומר:</u>
  - <u>להימנע משכפול קוד</u>
  - <u>להימנע מבדיקות טיפוס והמרות מיותרות (רמז: להיעזר בפולימורפיזם!)</u>

נתונות 2 ההיררכיות הבאות עבור זירות, ומתחרים.
עליכם להחליט לבד לגבי כל ישות מה מייצג אותה: מחלקה, מחלקה מופשטת או ממשק.
<u>שימו לב! הקווים בדיאגרמה אינם מייצגים בהכרח הורשה או קשר ישיר אחר.</u>





1. זירות: package game.arenas

1.1. Arena – ישות המתארת זירה.

1.1.1. Package: game.arenas

1.1.2. Fields:

- activeRacers: ArrayList<Racer>
- completedRacers: ArrayList<Racer>
- FRICTION: final double
- MAX_RACERS: final int
- MIN_Y_GAP: final static int
- length: double // x value of finish line.

1.1.3. Defaults:

- MIN_Y_GAP: 10

1.1.4. Constructors:

- Arena(double length, int maxRacers, double friction)

1.1.5. Methods

- addRacer(Racer newRacer): void

זורקת חריגה במידה ויש שגיאה בהוספת מתחרה:
אם הסוג אינו נכון (RacerTypeExeption)
אם אין מקום למתחרים נוספים (RacerLimitException))

- initRace(): void

על המתודה לקבוע לכל מתחרה מה היא נק' ההתחלה שלו. (בהמשך – יותאם לממשק הגרפי).
ערך X התחלתי יהיה 0.
ערך Y התחלתי יחושב ע"פ מרווח קבוע בין מתחרים וכמות המתחרים הפעילים בזירה, החל מ-0 לראשון.
המתודה מפעילה ()initRace של כל מתחרה עם קואורדינטות של נק' התחלה וסוף.

- hasActiveRacers(): boolean
- playTurn(): void

מפעילה את פונקציה התזוזה של כל המתחרים. אחרי כל תור יש לוודא כי השחקים הפעילים\שסיימו מעודכנים.

- crossFinishLine(Racer racer): void
- showResults(): void

מדפיסה את תוצאות המירוץ.

1.2. AerialArena

1.2.1. Package: game.arenas.air

1.2.2. Fields:

- DEFAULT_FRICTION: final static double
- DEFAULT_MAX_RACERS: final static int
- DEFAULT_LENGTH: final static int
- vision: enum Vision (***CLOUDS,SUNNY,FOG***)
- weather: enum Weather (***DRY,RAIN,SNOW***)
- height: enum Height (***LOW,MEDIUM,HIGH***)
- wind: enum Wind (***LOW,MEDIUM,HIGH***)

1.2.3. Defaults:

- DEFAULT_FRICTION: 0.4
- DEFAULT_MAX_RACERS: 6
- DEFAULT_LENGTH: 1500
- vision: ***SUNNY***
- weather: ***DRY***

- height: ***HIGH***
- wind: ***HIGH***

1.2.4. Constructors:
- AerialArena()
- AerialArena(double length, int maxRacers)

1.2.5. Methods:
- setWind(enum wind) / getWind()
- setVision(enum vision) / getVision()
- setHeight(enum height) / getHeight()
- setWeather(enum weather) / getWather()

NavalArena .1.3

1.3.1. Package: game.arenas.naval

1.3.2. Fields:
- DEFAULT_FRICTION: final static double
- DEFAULT_MAX_RACERS: final static int
- DEFAULT_LENGTH: final static int
- water : enum Water (***SALTED,SWEET***)
- surface: enum WaterSurface (***FLAT,WAVY***)
- Body: enum Body (***SEA,LAKE,RIVER,OCEAN***)

1.3.3. Defaults:
- DEFAULT_FRICTION: 0.7
- DEFAULT_MAX_RACERS: 5
- DEFAULT_LENGTH: 1000
- water: ***SWEET***
- surface: ***FLAT***
- body: ***LAKE***

1.3.4. Constructors:
- NavalArena()
- NavalArena(double length, int maxRacers)

1.3.5. Methods:
- setWater(enum water) / getWater()
- setSurface(enum surface) / getSurface()
- setBody(enum body) / getBody()

LandArena .1.4

1.4.1. Package: game.arenas.land

1.4.2. Fields:
- DEFAULT_FRICTION: final static double
- DEFAULT_MAX_RACERS: final static int
- DEFAULT_LENGTH: final static int
- coverage: enum Coverage (***SAND,GRASS,MUD***)
- surface: enum LandSurface (***FLAT,MOUNTAIN***)

1.4.3. Defaults:
- DEFAULT_FRICTION: 0.5
- DEFAULT_MAX_RACERS: 8
- DEFAULT_LENGTH: 800

- coverage: ***GRASS***
- surface: ***FLAT***

1.4.4. Constructors:
- LandArena()
- LandArena(double length, int maxRacers)

1.4.5. Methods:
- setCoverage(enum Coverage) / getCoverage()
- setSurface(enum surface) / getSurface()

2. מתחרים: package game.racers

2.1. **Racer** – ישות המתארת מתחרה

2.1.1. Package: game.racers

2.1.2. Fields:
- serialNumber: int

// לכל מתחרה יש מספר סידורי ייחודי, בסדר עולה.

- name: String
- currentLocation: Point
- finish: Point
- arena: Arena
- maxSpeed: double
- acceleration: double
- currentSpeed: double
- failureProbability: double // Chance to break down
- color: enum Color (***RED,GREEN,BLUE,BLACK,YELLOW***)
- mishap: Mishap

2.1.3. Defaults: none

2.1.4. Constructors:
- Racer(String name, double maxSpeed, double acceleration, <enum> color)

2.1.5. Methods:
- initRace(Arena arena, Point start, Point finish): void
- move(double friction): Point

*// accelerate if not at top speed: currSpeed += acceleration*friction.*
// move forward: currLocation.x += currSpeed (y is always 0 for now)
// has a chance for failure (see section 4.2)
// returns new location

- describeSpecific(): String

מחזירה מחזרות המייצגת את הפרטים הספציפיים לסוגים השונים (מספר גלגלים, גזע וכו').

- describeRacer(): String

מחזירה מחזרות המייצגת את המתחרה, ומכילה את כל הפרטים הבסיסיים של מתחרה (שם, מספר וכו'), וכן את הפרטים הספציפיים לסוגים השונים (מספר גלגלים, גזע וכו').

- introduce(): void

מדפיסה את סוג המתחרה ואת הפרטים שלו.

- className(): String

- hasMishap(): boolean
- getters and setters for each field

Wheeled .2.2

2.2.1. Package: game.racers
2.2.2. Fields:
- numOfWheels: int

2.2.3. Constructors:
- Wheeled(int numOfWheels)
- Wheeled()

2.2.4. Methods:
- describeSpecific(): String
- getter and setter for numOfWheels

AerialRacer .2.3

2.3.1. Package: game.racers.air
2.3.2. Fields: none
2.3.3. Defaults: none
2.3.4. Constructors: none
2.3.5. Methods: none

Airplane .2.4

2.4.1. Package: game.racers.air
2.4.2. Fields:
- CLASS_NAME: static final String
- DEFAULT_WHEELS: static final int
- DEFAULT_MAX_SPEED: static final double
- DEFAULT_ACCELERATION: static final double
- DEFAULT_color: static final Color
- wheeled: Wheeled

2.4.3. Defaults:
- CLASS_NAME: "Airplane"
- DEFAULT_MAX_SPEED: 885
- DEFAULT_ACCELERATION: 100
- DEFAULT_WHEELS: 3
- DEFAULT_color: ***BLACK***
- name: “Airplane #<serialNumber>

במידה ולא הוכנס שם, יש להכניס את המספר הסידורי של המתחרה במקום <> במחרוזת הנ"ל
למשל: Airplane #12

2.4.4. Constructors:
- Airplane()
- Airplane(String name, double maxSpeed, double acceleration, Color color, int numOfWheels)

2.4.5. Methods: none

Helicopter .2.5

2.5.1. Package: game.racers.air
2.5.2. Fields:
- CLASS_NAME: static final String

- DEFAULT_MAX_SPEED: static final double
- DEFAULT_ACCELERATION: static final double
- DEFAULT_color: static final Color

2.5.3. Defaults:
- CLASS_NAME: "Helicopter"
- DEFAULT_MAX_SPEED: 400
- DEFAULT_ACCELERATION: 50
- DEFAULT_color: ***BLUE***
- name: “Helicopter #<serialNumber>

במידה ולא הוכנס שם, יש להכניס את המספר הסידורי של המתחרה במקום <> במחרוזת הנ"ל
למשל: Helicopter #12

2.5.4. Constructors:
- Helicopter()
- Helicopter(String name, double maxSpeed, double acceleration, Color color)

2.5.5. Methods: none

2.6. **LandRacer** – ישות המייצגת מתחרה קרקע.

2.6.1. Package: game.racers.land
2.6.2. Fields: none
2.6.3. Defaults: none
2.6.4. Constructors: none
2.6.5. Methods: none

2.7. **Car**

2.7.1. Package: game.racers.land
2.7.2. Fields:
- CLASS_NAME: static final String
- DEFAULT_WHEELS: static final int
- DEFAULT_MAX_SPEED: static final double
- DEFAULT_ACCELERATION: static final double
- DEFAULT_color: static final Color
- wheeled: Wheeled
- engine: enum Engine (***FOURSTROKE, VTYPE, STRAIGHT, BOXER, ROTARY***)

2.7.3. Defaults:
- CLASS_NAME: "Car"
- DEFAULT_WHEELS: 4
- color: ***RED***
- engine: ***FOURSTROKE***
- DEFAULT_MAX_SPEED: 400
- DEFAULT_ACCELERATION: 20
- name: “Car #<serialNumber>

במידה ולא הוכנס שם, יש להכניס את המספר הסידורי של המתחרה במקום <> במחרוזת הנ"ל
למשל: Car #12

2.7.4. Constructors:
- Car()
- Car(String name, double maxSpeed, double acceleration, Color color, int numOfWheels)

2.7.5. Methods:

- getter and setter for engine

2.8. **Bicycle**

2.8.1. Package: game.racers.land

2.8.2. Fields:
- CLASS_NAME: static final String
- DEFAULT_WHEELS: static final int
- DEFAULT_MAX_SPEED: static final double
- DEFAULT_ACCELERATION: static final double
- DEFAULT_color: static final Color
- wheeled: Wheeled
- type: enum BicycleType (***MOUNTAIN,HYBRID,CRUISER,ROAD***)

2.8.3. Defaults:
- CLASS_NAME: "Bicycle"
- DEFAULT_WHEELS: 2
- DEFAULT_color: ***GREEN***
- type: ***MOUNTAIN***
- DEFAULT_MAX_SPEED: 270
- DEFAULT_ACCELERATION: 10
- name: “Bicycle #<serialNumber>

במידה ולא הוכנס שם, יש להכניס את המספר הסידורי של המתחרה במקום <> במחרוזת הנ"ל
למשל: Bicycle #12

2.8.4. Constructors:
- Bicycle(String name, double maxSpeed, double acceleration, Color color, int numOfWheels)
- Bicycle()

2.8.5. Methods:
- getter and setter for type

2.9. **Horse**

2.9.1. Package: game.racers.land

2.9.2. Fields:
- CLASS_NAME: static final String
- DEFAULT_MAX_SPEED: static final double
- DEFAULT_ACCELERATION: static final double
- DEFAULT_color: static final Color
- breed: enum Breed (***THOROUGHBRED,STANDARDBRED,MORGAN,FRIESIAN***)

2.9.3. Defaults:
- CLASS_NAME: "Horse"
- DEFAULT_MAX_SPEED: 50
- DEFAULT_ACCELERATION: 3
- DEFAULT_color: ***BLACK***
- breed: ***THOROUGHBRED***
- name: “Horse #<serialNumber>

במידה ולא הוכנס שם, יש להכניס את המספר הסידורי של המתחרה במקום <> במחרוזת הנ"ל
למשל: Horse #12

2.9.4. Constructors:

- Horse(String name, double maxSpeed, double acceleration, Color color)
- Horse()

2.9.5. Methods:

- getter and setter for breed

NavalRacer .2.10

2.10.1. Package: game.racers.naval
2.10.2. Fields: none
2.10.3. Defaults: none
2.10.4. Constructors: none
2.10.5. Methods: none

RowBoat .2.11

2.11.1. Package: game.racers.land.naval
2.11.2. Fields:

- CLASS_NAME: static final String
- DEFAULT_MAX_SPEED: static final double
- DEFAULT_ACCELERATION: static final double
- DEFAULT_color: static final Color
- type: enum BoatType (***SKULLING,SWEEP***)
- team: enum Team (***SINGLE,DOUBLE,QUAD,EIGHT***)

2.11.3. Defaults:

- CLASS_NAME: "RowBoat"
- DEFAULT_color: ***RED***
- DEFAULT_MAX_SPEED: 75
- DEFAULT_ACCELERATION: 10
- type: ***SKULLING***
- team: ***DOUBLE***
- name: “RowBoat #<serialNumber>

במידה ולא הוכנס שם, יש להכניס את המספר הסידורי של המתחרה במקום <> במחרוזת הנ"ל
למשל: RowBoat #12

2.11.4. Constructors:

- RowBoat(String name, double maxSpeed, double acceleration, Color color)
- RowBoat()

2.11.5. Methods:

- getters and setters for type and team

SpeedBoat .2.12

2.12.1. Package: game.racers.land.naval
2.12.2. Fields:

- CLASS_NAME: static final String
- DEFAULT_MAX_SPEED: static final double
- DEFAULT_ACCELERATION: static final double
- DEFAULT_color: static final Color
- type: enum BoatType (***SKULLING,SWEEP***)
- team: enum Team (***SINGLE,DOUBLE,QUAD,EIGHT***)

2.12.3. Defaults:

- CLASS_NAME: "SpeedBoat"
- DEFAULT_color: ***RED***

- type: ***SKULLING***
- team: ***DOUBLE***
- DEFAULT_MAX_SPEED: 170
- DEFAULT_ACCELERATION: 5

2.12.4. Constructors:

- SpeedBoat(String name, double maxSpeed, double acceleration, Color color)
- SpeedBoat()

2.12.5. Methods:

- getters and setters for type and team

3. מארגן מתרות: package factory – קוד שאחראי על בניית הזירה ומתחרים.

3.1. **RaceBuilder** – מחלקה בונה זירה עם מתחרים. יש להשתמש בטעינה דינאמית (**java reflection**)! המחלקה הזאת צריכה לממש תבנית **Singleton** . יש לדמות קליטת קלט מהמשתמש (סוג הזירה, מאפיינים שלה, כמות המתחרים, וכד') ע"י העברת ארגומנטים לבנאים של זירות ומתחרים (ראה דוגמת הרצה ב-Program.java). בעבודה הבאה, הקלט ייקלט דרך **GUI**.

3.1.1. Package: utilities

3.1.2. Fields:

- Instance: static RaceBuilder
- classLoader: ClassLoader
- classObject: Class<?>
- constructor: Constructor<?>

3.1.3. Methods:

- buildArena(String arenaType, double length, int maxRacers): Arena
- buildRacer(String racerType, String name, **double** maxSpeed, **double** acceleration, utilities.EnumContainer.Color color): Racer
- buildWheeledRacer(String racerType, String name, **double** maxSpeed, **double** acceleration, utilities.EnumContainer.Color color, int numOfWheels): Racer
- מתודות עזר – לפי שיקולכם

4. מחלקות עזר: package utulities

4.1. **Program** – המחלקה הראשית.

4.2. **Point** – מגדירה מיקום על ציר דו מימדי. הצרים נעים בין הערכים הבאים (יש לשמור את ערכים אלו **כקבועים במחלקה**).

4.2.1. Fields:

- MAX_X: static final int
- MIN_X: static final int
- MAX_Y: static final int
- MIN_Y: static final int
- double x
- double y

4.2.2. Defaults:

- MAX_X: 1000000
- MIN_X: 0
- MAX_Y: 800
- MIN_Y: 0

4.2.3. Constructors:

- Point(double, double)

- Point()
- Point(Point)

4.2.4. Methods:

- toString()
- getters and setters

4.3. **Fate –** מחלקת שירות המגרילה אירועים המשפיעים על תוצאת התחרות, כגון: תקלות, תאוצה, וכד'.
התוצאות הגרלה צריכות להיחשב ע"פ נתוני הסתברות של המתחרה. ניתן להשתמש בקוד שפורסם או לממש לפי דרישות הבאות:

4.3.1. Package: utilities

4.3.2. Methods:

- setSeed(int seed): void
- generateFixable(): boolean
- generateReduction(): float
- generateTurns(): int
- breakDown(): boolean
- generateMishap(): Mishap

4.4. **Mishap** – מחלקה המייצגת תקלה.
כל תקלה מכילה 3 פרמטרים – האם ניתן לתקן, הזמן לתיקון (בתורות), וההשפעה על התאוצה של המתחרה. התקלה יכולה לקרות בזמן תזוזה באופן הבא:
בכל פעם שהמתחרה זז (במטודה move) יש לבדוק האם קיימת תקלה. (תקלה שניתן לתקן ומספר התורות בה הוא 0 – נחשבת כאילו אין תקלה וניתן לדרוס אותה בnull). אם אין – יש להגריל אחת חדשה, ע"י פניה למטודה brakeDown במחלקה Fate. אם חזר "אמת", יש לייצר תקלה חדשה ע"י המטודה generateMishap תחת Fate.
לאחר מכן, אם ישנה תקלה (קיימת או חדשה) – יש להכפיל את התאוצה בגורם ההאטה (reductionFactor), ולהקטין את הזמן הנותר לתיקון.

4.4.1. Package: utilities

4.4.2. Fields:

- fixable :boolean
- reductionFactor: double
- turnsToFix: int

4.4.3. Constructors:

- Mishap(boolean fixable, int turnsToFix, double reductionFactor)

4.4.4. Methods:

- nextTurn(): void

מקטינה את כמות התורים לתיקון (במידה והתקלה ניתן לתיקון)

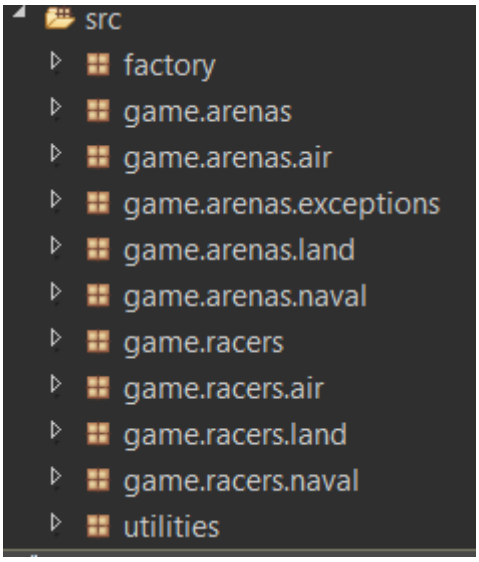
- getters and setters
- toString example: (false, 3 ,0.90)

[you can use DecimalFormat("0.00").format(reductionFactor) to limit digits of double]

.

הערות כלליות:

דוגמא להרצת תכנית (הרכבת זירה, יצירת מתחרים והפעלת מרוץ) נמצאת תחת הקובץ Program.java.
אנא שימו לב – יש שם שורה שמעלה חריגה – זה מכוון. אתם אמורים לקבל את אותו השגיאה, אך ייתכן שהשורות שלה יופיעו במיקום קצת שונה (ואף לא ברצף).
מומלץ מאוד לפתח לאורך ולא לרוחב – כלומר לממש מרוץ שלם של מטוסים שעובד כמו שצריך, ורק אז לעבור לסוגי זירות\מתחרים נוספים.
כאשר אובייקט מועבר בין מחלקות, יש להבין האם עלינו לשמור *reference* או ליצור עותק חדש, כדי למנוע שינויים לא רצויים מבחוץ.
ניתן לראות בתמונה למטה את המבנה הכללי של תכנית מבחינת חבילות.



עבודה נעימה!!!