<u>תרגול מס' 9</u>

הורשה ופולימורפיזם - המתודות toString ו - equals, בירור סוג העצם, המרה כלפי מטה (downcasting)

דגשים לפתרון:

א. הוסיפו לפרויקט package בשם T9 ולתוכו הוסיפו קובץ מחלקה בשם T9 הכולל main.

<u>תרגיל 1 - תרגיל כיתה</u>

נמשיך לעבוד עם שלוש המחלקות מתרגול 8 (Question, OpenQuestion, ClosedQuestion).

ניתן להוריד את קבצי הקוד המלאים מה- mama ולהעתיק אותם אל חבילה T9.

בצעו שינויים בקבצים המועתקים לפי ההנחיות שבהמשך.

א. במחלקה **שאלה** (**Question**)

1. דרסו את המתודה ***equals*** כך שתחזיר true אם האובייקט שהתקבל כפרמטר הוא מסוג שאלה וערכי התכונות שלו זהים לערכי התכונות של העצם הנוכחי. אחרת, יוחזר false.
2. דרסו את המתודה ***toString*** כך שתחזיר מחרוזת המכילה את פרטי העצם באותה תבנית כמו זו שהמתודה ***show*** מדפיסה. למשל עבור **שאלה** שאינה חובה עם מזהה 1 והכותרת "Write your favourite country's name" תוחזר המחרוזת:

"1. Write your favourite country's name (not mandatory)"

ב. בפונקציה הראשית (בקובץ T9.java) כתבו הוראות מתאימות:

1. צרו עצם של **שאלה** שישמר במשתנה הפנייה q1.
מזהה השאלה הוא 1, הכותרת שלה היא "Wriet your favourite country's name" והיא אינה שאלת חובה.
2. צרו עותק של q1 שישמר במשתנה הפנייה q2 .
3. הדפיסו את פרטי השאלה q1 על המסך באמצעות זימון מפורש של המתודה ***toString***
4. הדפיסו את פרטי השאלה q2 על המסך ללא זימון מפורש של ***toString*** (ב - syso כתבו את q2 בלבד)
5. עבור q1 זמנו את המתודה ***equals*** שתקבל כפרמטר את q2. הדפיסו את תוצאת הזימון (אמור להיות true)
6. עבור q2 זמנו את המתודה ***equals*** שתקבל כפרמטר את q1. הדפיסו את תוצאת הזימון (אמור להיות true)
7. עבור q2 זמנו את המתודה ***equals*** שתקבל כפרמטר את q2. הדפיסו את תוצאת הזימון (אמור להיות true)
8. עדכנו את המזהה של q2 שיהיה 2
9. עבור q2 זמנו את המתודה ***equals*** שתקבל כפרמטר את q1. הדפיסו את תוצאת הזימון (אמור להיות false)

ג. במחלקה **שאלה פתוחה** (**OpenQuestion**)

1. לצורך המחשת מושגים הקשורים בתרגול הזה נדרש למחוק את הדריסה של המתודה ***setTitle*** שביצענו בתרגול הקודם.
   וודאו שגם בבנאי אין עדכון של התכונה title באופן שונה מזה המבוצע במחלקת האב
2. דרסו את המתודה ***equals*** כך שתחזיר true אם האובייקט שהתקבל כפרמטר הוא מסוג **שאלה פתוחה** וערכי התכונות שלו זהים לערכי התכונות של העצם הנוכחי. אחרת, יוחזר false. **חשבו**: האם ניתן לקצר בקוד של המתודה הזו?
3. דרסו את המתודה ***toString*** כך שתחזיר מחרוזת המכילה את פרטי העצם באותה תבנית כמו זו שהמתודה ***show*** מדפיסה. למשל, עבור **שאלת** חובה **פתוחה** עם מזהה 3, כותרת "Write about your last vacation" ומענה של מספר שורות תוחזר המחרוזת:

   3. Write about your last vacation (mandatory) **in several lines**

ד. הוסיפו לפונקציה הראשית (בקובץ T9.java) הוראות מתאימות:

1. צרו עצם של **שאלה פתוחה** שישמר במשתנה הפנייה q3.
   מזהה השאלה הוא 3, הכותרת "Write about your last vacation" והיא שאלת חובה עם מספר שורות.
2. הדפיסו את פרטי השאלה q3 על המסך.
3. צרו עותק של q3 שישמר במשתנה הפנייה q4 .
4. עבור q3 זמנו את המתודה ***equals*** שתקבל כפרמטר את q4. הדפיסו את תוצאת הזימון (אמור להיות true)
5. עדכנו את q4 כך שלא תהיה שאלת חובה, והמענה עליה יהיה בשורה בודדת
6. עבור q3 זמנו שוב את המתודה ***equals*** שתקבל כפרמטר את q4. הדפיסו את תוצאת הזימון (אמור להיות false)
7. עדכנו את המזהה (id) והכותרת (title) של q4 כך שערכיהן יהיו זהים למזהה והכותרת של q1
8. עבור q4 זמנו את המתודה ***equals*** שתקבל כפרמטר את **q1**. הדפיסו את תוצאת הזימון (אמור להיות false. מדוע?)
9. עבור **q1** זמנו את המתודה ***equals*** שתקבל כפרמטר את **q4**. הדפיסו את תוצאת הזימון (אמור להיות true. מדוע?)

ו. הוסיפו מעל הפונקציה הראשית (בקובץ T9.java) פונקציה חדשה בשם mandatorySingleLine המקבלת כפרמטר את qs, מערך של שאלות.
הפונקציה תדפיס על המסך את הפרטים של העצמים ב - qs שהם שאלות חובה פתוחות והמענה עליהן הוא בשורה בודדת.

ז. בפונקציה הראשית צרו מערך של 4 שאלות שישמר במשתנה ששמו questions.
בצעו השמה של משתני ההפניה q1 ועד q4 לתאי המערך questions.
זמנו את הפעולה mandatorySingleLine והעבירו אליה את questions כפרמטר.
הפרטים של כמה מתוך ארבע השאלות מודפסים?

תרגיל 2 - תרגיל כיתה

נתונות המחלקות הבאות: SpaceShip, TransportShip, BattleShip, Fighter, Bomber

הורידו את קבצי המחלקות מאתר הקורס, שמרו אותם אצלכם וענו על השאלות הבאות:

1. שרטטו את עץ ההורשה המבטא את הקשר בין המחלקות
2. צרו פרוייקט או חבילה חדשה בשם T9 וצרפו אליה את קבצי המחלקות.
3. **במחלקה SpaceShip** דרסו את הפעולה equals כך שתחזיר true במידה ושני העצמים הם מאותו הטיפוס ובעלי שם זהה. בכל מקרה אחר יוחזר false. כתבו את הפעולה כך שתתאים לכל המחלקות בפרוייקט ללא צורך בדריסה.
4. הוסיפו לפרוייקט מחלקה TestMain ובה פעולה ראשית. במחלקה הזו כתבו את הפעולות הבאות:
   a. כתבו פעולה בשם allDifference. הפעולה תקבל מערך של SpaceShip ותחזיר true אם כל העצמים שבמערך שונים זה מזה.
   b. כתבו פעולה בשם totalMaintenance. הפעולה תקבל מערך של SpaceShip ותחזיר את סכום התחזוקה הכולל עבור כל רכבי החלל שבמערך.
   c. כתבו פעולה בשם totalFireCost. הפעולה תקבל מערך של SpaceShip ותחזיר את עלות החימוש המצטבר של כל רכבי החלל.
5. בפעולה הראשית צרו את האובייקטים הבאים והכניסו אותם לתוך מערך
   a. ספינת חלל SpaceShip בשם "a1" בעלת מהירות מקסימלית 100
   b. ספינת חלל SpaceShip בשם "a2" בעלת מהירות מקסימלית 200
   c. ספינת תובלה TransportShip בשם "a1", בעלת מהירות מקסימלית 85, עם תפוסת מטען - 50 ותפוסת נוסעים 75
   d. ספינת לוחמים Fighter בשם "b1" בעלת מהירות מקסימלית 120, עם כלי נשק - 30, כוח אש 330, ומספר לוחמים 18

e. ספינת הפצצה Bomber בשם "f1" בעלת מהירות מקסימלית 160, עם כלי נשק - 55, כוח אש 700, ומספר משגרים 10

6. בצעו את ההוראות הבאות:

a. הדפיסו את כל ערכי האובייקטים במערך

b. בידקו האם כל האובייקטים במערך שונים זה מזה

c. הציגו את עלות התחזוקה הכוללת של כל ספינות החלל

d. הציגו את עלות החימוש הכוללת של ספינות החלל

```
public class SpaceShip
{
	protected String name;
	protected double maxSpeed;


	public SpaceShip(String name, double maxSpeed) {
		setName(name);
		this.maxSpeed = maxSpeed;
	}

	public String getName() {
		return this.name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public void setMaxSpeed(double maxSpeed) {
		this.maxSpeed = maxSpeed;
	}

	public double getMaxSpeed() {
		return maxSpeed;
	}

	public int getAnnualMaintenanceCost() {
		return 0;
	}

	@Override
	public String toString() {
		return  this.getClass() +"\n" +
				"\tName=" + this.getName() + "\n" +
				"\tMaxSpeed=" + this.maxSpeed + "\n" +
				"\tAnnualMaintenanceCost=" + this.getAnnualMaintenanceCost();
```

```
	}

}
```

```
public class TransportShip extends SpaceShip{

	private final int BASE_ANNUAL_MAINTENANCE_COST = 3000;
	private int cargoCapacity;
	private int passengerCapacity;

	public TransportShip(String name, double maxSpeed, int cargoCapacity, int passengerCapacity)
	{
		super(name, maxSpeed);
		this.cargoCapacity = cargoCapacity;
		this.passengerCapacity = passengerCapacity;
	}
	@Override
	public int getAnnualMaintenanceCost() {
		return BASE_ANNUAL_MAINTENANCE_COST +
				5 * this.cargoCapacity +
				3 * this.passengerCapacity;
	}

	@Override
	public String toString() {
		String st = super.toString() +
				"\n\tCargoCapacity=" + this.cargoCapacity +
				"\n\tPassengerCapacity=" + this.passengerCapacity;
		return st;
	}

}
```

```
public class BattleShip extends SpaceShip{

	protected final int BASE_FIREPOWER = 10;
	protected int firePower;
	protected int weapons;

	public BattleShip(String name, double maxSpeed, int weapons, int firePower)
	{
		super(name, maxSpeed);
		this.weapons = weapons;
		this.firePower = firePower;
	}

	public int getBASE_FIREPOWER() {
		return BASE_FIREPOWER;
	}

	public int getFirePower() {
		return  firePower;
	}
```

```
	public int getWeapons() {
		return weapons;
	}

	public void setFirePower(int firePower) {
		this.firePower = firePower;
	}

	public void setWeapons(int weapons) {
		this.weapons = weapons;
	}

	public int getAnnualArmenantCost() {
		return  weapons*(BASE_FIREPOWER + firePower);
	}

	@Override
	public int getAnnualMaintenanceCost() {
		int cost =  super.getAnnualMaintenanceCost();
		cost += (int)weapons*0.8;
		return cost;
	}

	@Override
	public String toString() {
		String st = super.toString() +
				"\n\tWeapons=" + this.getWeapons()+
				"\n\tfirePower=" + this.getFirePower();
		return st;
	}
}
```

```
public class Fighter extends BattleShip
{

	private final int COST_PER_FIGHTER = 2500;
	protected int numOfFighters;

	public Fighter(String name, double maxSpeed, int weapons, int firePower, int numOfFighters)
	{
		super(name, maxSpeed, weapons, firePower);
		this.numOfFighters=numOfFighters;

	}

	public int getNumOfFighters() {
		return numOfFighters;
	}

	public void setNumOfFighters(int numOfFighters) {
		this.numOfFighters = numOfFighters;
	}

	public int getCOST_PER_FIGHTER() {
		return COST_PER_FIGHTER;
```

```
	}


	@Override
	public int getAnnualArmenantCost() {
		return super.getAnnualArmenantCost()*numOfFighters;
	}


	@Override
	public int getAnnualMaintenanceCost() {

		return  super.getAnnualMaintenanceCost() +
				getCOST_PER_FIGHTER()*numOfFighters;
	}

	@Override
	public String toString() {
		return  super.toString() +
				"\n\tNumberOfFighters=" + this.numOfFighters;
	}

}
```

```
public class Bomber extends BattleShip{

	private final int BASE_ANNUAL_MAINTENANCE_COST = 5000;
	private int numOfLaunchers;

	public Bomber(String name, double maxSpeed, int weapons, int firePower, int numOfLaunchers)
	{
		super(name, maxSpeed, weapons, firePower);
		this.numOfLaunchers=numOfLaunchers;

	}

	public int getBaseAnnualMaintenanceCost()
	{
		return BASE_ANNUAL_MAINTENANCE_COST;
	}

	public int getNumOfLaunchers() {
		return numOfLaunchers;
	}


	public void setNumOfLaunchers(int numOfLaunchers) {
		this.numOfLaunchers = numOfLaunchers;
	}


	@Override
	public int getAnnualArmenantCost() {
		return super.getAnnualArmenantCost()+numOfLaunchers*firePower;
	}
```

```
	@Override
	public int getAnnualMaintenanceCost() {
		int cost =   super.getAnnualMaintenanceCost();
		return cost + (int)(getBaseAnnualMaintenanceCost() *(10-numOfLaunchers/10.0));

	}

	@Override
	public String toString() {
		return  super.toString() +
				"\n\tNumberOfLaunchers=" + this.numOfLaunchers;
	}

}
```