**Lab 06: Rocket Analysis**

**Muhammad Rizwan Khalid**

**180459**

**BSCS-6A**

**Implementation of Problem: -**

We have to apply the object-oriented concepts to model the rocket in java. We have to analyze the basic entities and attributes of the rocket. The modelling of varies from person to person because OOP concept depend upon objects, its attributes and its methods.

I figured it out in my own way. I made the separate class of rocket, engine, nozzle and propellant. Since we know that rocket contains engine and engine contains propellant and nozzle.

In the class of rocket, I made the attributes and some methods and created an object of class engine in the rocket class. In engine class, I made the objects of propellant and nozzle class.

In propellant class, I put all the methods and attributes related to fuel because propellant carries fuel and engine ignites fuel.

In nozzle class, I just made the attributes because nozzle is only responsible for emission of gasses.

**Solution to Problem:**

package pkgcase;

import java.util.\*;

public class Case {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

String name;

double distance, time;

System.out.println("Here is the Rocket!! By default fuel is full.");

System.out.print("Enter the name of Rocket: ");

name = input.next();

System.out.print("Enter the distance(kms) to your destination: ");

distance = input.nextDouble();

Rocket rocket = new Rocket(name,distance);

rocket.engine.calculateThrust();

System.out.println(rocket.timeRequired()+" Hours will be taken to reach destination.");

System.out.print("Enter time to get distace after that time: ");

time = input.nextDouble();

System.out.println("You have covered "+rocket.distanceAtTime(time)+" kms");

System.out.println("Your remaining distance is "+rocket.getDistance()+" kms");

System.out.println(rocket.timeRequired()+" Hours are remaining to reach destination.");

System.out.println("Initial fuel was "+rocket.engine.propellent.getFuel());

System.out.println(rocket.engine.propellent.fuelRequired(distance)+"Litters fuel requird to travel remaining distance");

}

}

class Rocket{

private String name;

private final double weight = 100000; // 1 lac kgs excluding fuel

private final double velocity = 11.2; //kms-1

private double distance; // km

Engine engine = new Engine();

public Rocket() {

name = null;

distance = 0;

System.out.println("Rocket Has been created.");

}

public Rocket(String name, double distance) {

this.name = name;

this.distance = distance;

System.out.println("Rocket Has been created.");

}

public void setName(String name){

this.name = name;

}

public void setDistance(double distance){

this.distance = distance;

}

public String getName(){

return name;

}

public double getDistance(){

return distance;

}

public double timeRequired(){ // Time taken (hours) to cover distanceTravel

double time = (this.distance / (this.velocity \* 3600)) ;

return time;

}

public double timeRequired(double distance){ //Time taken (hours) to cover distance

double time = (distance / (this.velocity \* 3600));

return time;

}

public double distanceAtTime(double time){

double res = (this.velocity \* time \* 3600);

this.distance = this.distance - res;

return res;

}

}

class Engine{

Propellent propellent = new Propellent();

Nozzle nozzle = new Nozzle();

private double ratio; //thrust-to-weight ratio.

public Engine() {

System.out.println("Rocket Engine is all Set.");

}

public void calculateThrust(){

System.out.println("Net upward thrust equals change in momentum!!");

}

public void setRatio(double ratio){

this.ratio = ratio;

}

public double getRatio(){

return this.ratio;

}

}

class Nozzle{

private double size;

private double specificImpulse;

private double exhaustPressure;

public Nozzle() {

specificImpulse = 0;

exhaustPressure = 0;

size = 0;

}

public double getSpecificImpulse(){

return this.specificImpulse;

}

public void setSpecificImpulse(double specificImpulse){

this.specificImpulse = specificImpulse;

}

public double getExhaustPressure(){

return this.exhaustPressure;

}

public void setExhaustPressure(double exhaustPressure){

this.exhaustPressure = exhaustPressure;

}

public double getSize(){

return this.size;

}

public void setSize(double size){

this.size = size;

}

}

class Propellent{

private double fuel;

private static double fuelCapacity = 2500000; // 2.5 million litters of fuel

private static double combustionRate = 5000; // litters of fuel per second

public Propellent() {

fuel = fuelCapacity;

}

public Propellent(double fuel) {

if (fuel > fuelCapacity){

System.err.println("Fuel Overflow!!");

this.fuel = fuelCapacity;

}

else{

this.fuel = fuel;

}

}

public void setFuel(double fuel){

if (fuel > fuelCapacity){

System.err.println("Fuel Overflow!!");

this.fuel = fuelCapacity;

}

else{

this.fuel = fuel;

}

}

public double getFuel(){

return this.fuel;

}

public double getFuelCapacity(){

return Propellent.fuelCapacity;

}

public double getCombustionRate(){

return Propellent.combustionRate;

}

public double fuelRequired(double distance){

double time = distance / (11.2 \* 3600);

double res = (combustionRate \* time \* 3600);

return res;

}

public double fuelRemaining(double distance){

double fuelUsed = fuelRequired(distance);

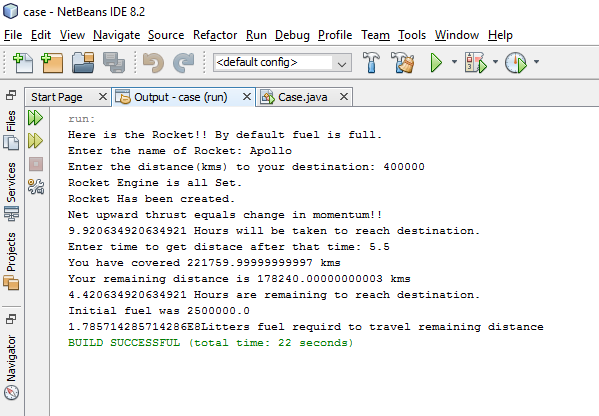
this.fuel = this.fuel - fuelUsed;

return this.fuel;

}

}

**Output:**



**P.T.O**

**UML Diagram:**

|  |
| --- |
| **Rocket** |
| -name: String  -distance: double  -velocity: double  -weight: double  engine: Engine  +<<constructor>> Rocket()  +<<constructor>> Rocket(name:String, distance:double) |
| +setName(name: String): void  +setDistance(distance: double): void  +getName(): String  +getDistance(): double  +timeRequired(): double  +timeRequired(distance: double): double  +distanceAtTime(time: double): double |

|  |
| --- |
| **Engine** |
| propellent: Propellent  nozzle: Nozzle  -ratio: double  <<constructor>> Engine() |
| +calculateThrust(): void  +setRatio(ratio: double): void  +getRatio(): double |

|  |
| --- |
| **Nozzle** |
| -size: double  -specificImpulse: double  -exhaustPressure: double  <<constructor>> Nozzle() |
| +setSize(size: double): void  +setSpecificImpulse(specificImpulse: double): void  +setExhaustPressute(exhaustPressure: double):void  +getSize(): double  +getSpecificImpulse(): double  +getExhaustPressure(): double |

|  |
| --- |
| **Propellent** |
| -fuel: double  -fuelCapacity  -combustionRate  <<constructor>> Propellent()  <<constructor>> Propellent(fuel: Fuel) |
| +setFuel(fuel: double): void  +getFuel(): double  +getFuelCapacity(): double  +getCombustionRate: double  +fuelRequired(distance: double): double  +fuelRemaining(distance: double): double |