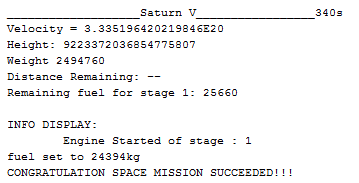
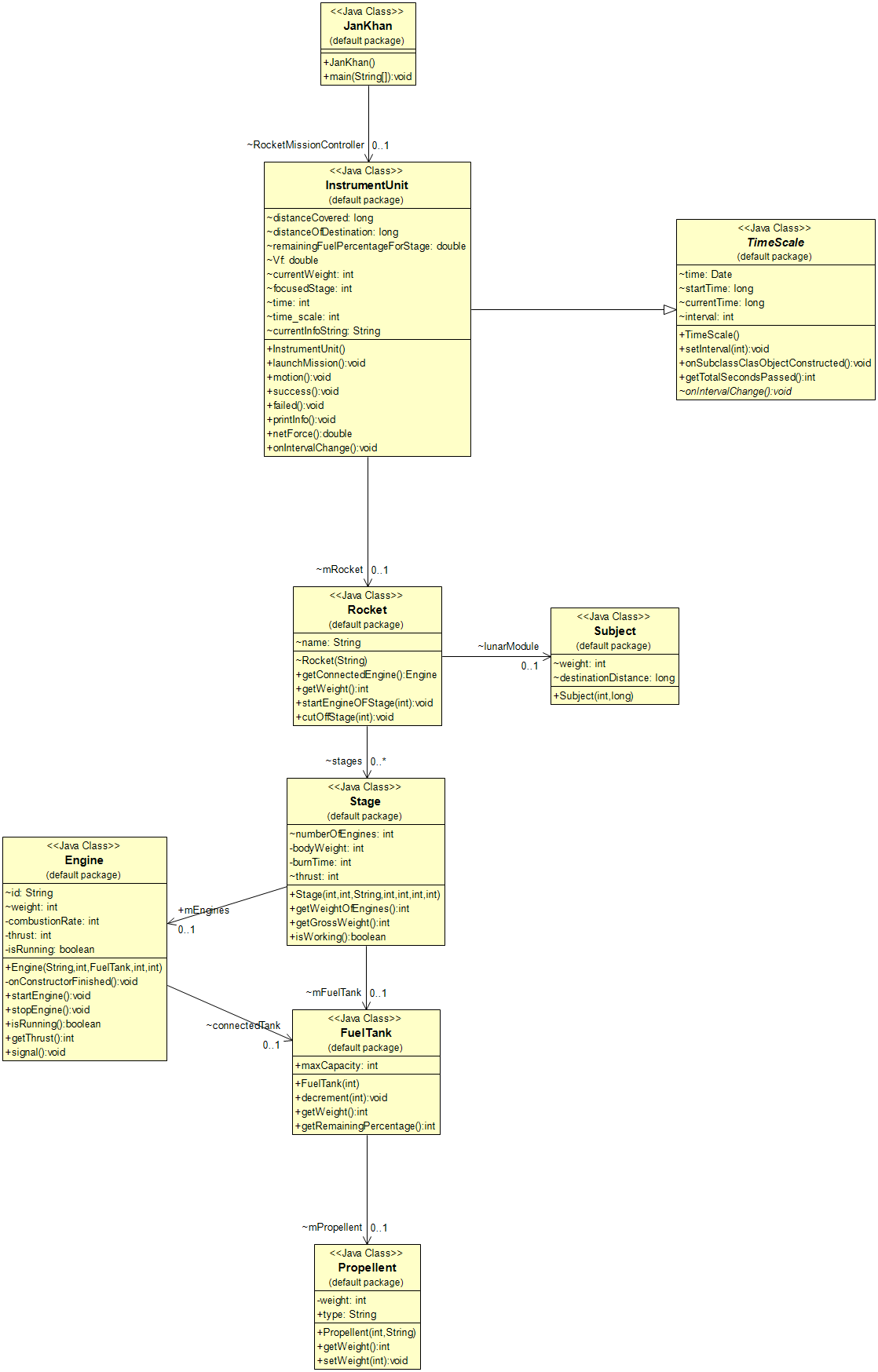
# ROCKET SCIENCE

## Problem Statement:

The objective is to simulate the space mission to moon with rocket Saturn V

* Saturn V is a three-stage rocket. The lunar module is fixed in the rocket and sent to moon.
* Every stage has its own fuel tank and a several engines.
* A stage is cut off from the rocket when its fuel combusts so that the total weight of the rocket is least as possible.
* Instrument unit controls the whole flight of the rocket and decides the tasks
* So, the goal is to simulate and track the whole flight of the rocket by displaying its velocity, height weight, distance remaining, remaining fuel and a piece of information which tells current situation of rocket or running processes.





# CODE



## Engine

1. public class Engine {
3. String id;
4. int weight;
5. private int combustionRate;
6. private int thrust;
7. public FuelTank connectedTank;
8. private boolean isRunning = false;



13. public Engine(String id, int weight, FuelTank tankReference, int combustionRate, int thrust) {
14. this.id = id;
15. this.weight = weight;
16. this.combustionRate = combustionRate;
17. this.connectedTank = tankReference;
18. this.thrust = thrust;
20. }
22. private void onConstructorFinished() {
24. }
26. public void startEngine() {
28. isRunning = true;
29. }
30. public void stopEngine() {
31. isRunning = false;
32. }
33. public boolean isRunning() {
34. return isRunning;
35. }
37. public int getThrust() {
38. if (isRunning) {
39. return thrust;
40. }
41. return 0;
42. }
44. public void signal() {
45. this.connectedTank.decrement(combustionRate);
46. }


50. }

## Engine

1. public class FuelTank {
3. private Propellent mPropellent;
4. public int maxCapacity;
6. public FuelTank(int maxCapacity) {
7. mPropellent = new Propellent(maxCapacity, "solid");
8. this.maxCapacity = maxCapacity;
10. }
11. public void decrement(int x) {
12. mPropellent.setWeight(mPropellent.getWeight() - x);
13. }
14. public int getWeight() {
15. return mPropellent.getWeight();
16. }
17. public int getRemainingPercentage() {
18. return ((getWeight()) / maxCapacity) \* 100;
19. }





26. }

## InstrumentUnit

1. public class InstrumentUnit extends TimeScale {
2. Rocket mRocket;
3. //all info....
4. long distanceCovered = 19999994;
5. long distanceOfDestination;
6. double remainingFuelPercentageForStage;
7. double Vf = 0;
8. int currentWeight;
9. int focusedStage = 1;
10. int time = 0;
11. int time\_scale = 1000;
12. String currentInfoString = "Mission Not Launched yet.";

15. public InstrumentUnit() {
16. this.mRocket = new Rocket("Saturn V");
17. super.setInterval(time\_scale);
18. Vf = 0;
19. this.distanceOfDestination = this.mRocket.lunarModule.destinationDistance;
20. this.currentInfoString = "Building Rocket";
22. }


26. public void launchMission() {

29. while (focusedStage < 3) {
30. time++;
31. mRocket.stages[focusedStage].mEngines.startEngine();
32. currentInfoString = "Engine Started of stage : " + focusedStage;
33. mRocket.getConnectedEngine().signal();
34. this.onIntervalChange();
35. if (mRocket.stages[focusedStage].mFuelTank.getWeight() < 0) {
36. mRocket.cutOffStage(focusedStage);
37. System.out.println("STAGE:" + focusedStage + " CUT OFF");
38. focusedStage++;
40. }
41. if (distanceCovered > this.distanceOfDestination) {
42. success();
43. }
45. printInfo();
46. }
47. System.out.println("ALL STAGES CUT OFF");
48. motion();




54. }
55. public void motion() {

58. while (true) {
59. onIntervalChange();
60. if (this.distanceCovered > this.distanceOfDestination) {
61. success();
63. } else if (this.distanceCovered <= 0) {
64. failed();
65. }
66. }


70. }
71. public void success() {
72. System.out.println("CONGRATULATION SPACE MISSION SUCCEEDED!!!");
73. }
74. public void failed() {
75. System.out.println("SORRY YOUR SPACE MISSION FAILED");
76. this.onIntervalChange();
77. }


81. public void printInfo() {
82. //clearConsole();
83. System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" + mRocket.name + "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" + time + "s");
84. System.out.println("Velocity = " + Vf);
85. System.out.println("Height: " + distanceCovered);
86. System.out.println("Weight " + mRocket.getWeight());
87. System.out.println("Distance Remaining: --");
88. int temp = mRocket.getConnectedEngine().connectedTank.getWeight();
89. System.out.println("Remaining fuel for stage " + focusedStage + ": " + temp);
90. temp = mRocket.getConnectedEngine().connectedTank.getRemainingPercentage();
91. temp = temp / 10;
92. for (int i = 0; i < temp; i++) {
93. System.out.printf("|");
94. }
95. for (int i = 0; i < 10 - temp; i++) {
96. System.out.printf(" ");
97. }
98. //System.out.printf("|\nTotal TIme elapsed : %d",super.getTotalSecondsPassed());
99. System.out.printf("\nINFO DISPLAY: \n\t%s\n", currentInfoString);



104. }




110. public double netForce() {
111. double thrust = mRocket.getConnectedEngine().getThrust();
112. double Moon = (9 \* 10e11) \* (mRocket.getWeight() \* 7.34e22) / Math.pow(distanceOfDestination - distanceCovered, 2);
113. double Earth = (9 \* 10e11) \* (mRocket.getWeight() \* 5.97e24) / Math.pow(distanceOfDestination + distanceCovered, 2);
114. return thrust + Moon + Earth;
115. }

118. public void onIntervalChange() {
120. if (mRocket.getConnectedEngine() != null) {
121. if (mRocket.getConnectedEngine().isRunning) {
123. //update info for accelaration
124. double Vi = Vf;
125. currentWeight = mRocket.getWeight();
126. double a = netForce() / currentWeight;
128. Vf = Vf + a \* 1;
130. double s = (Math.pow(Vf, 2) - Math.pow(Vi, 2)) / (2 \* a);
131. if (s > 0)
132. distanceCovered += s;
133. remainingFuelPercentageForStage = (mRocket.getConnectedEngine().connectedTank.getWeight() / mRocket.getConnectedEngine().connectedTank.maxCapacity) \* 100;
134. }
135. } else {
137. double a = netForce() / currentWeight;
138. double s = (Math.pow(Vf, 2) - Math.pow(Vf, 2)) / (2 \* a);
139. if (s > 0)
140. distanceCovered += s;
142. }
143. //        try{
144. //          printInfo();
145. //        }
146. //        catch(Exception e){
147. //
148. //        }
150. }


154. }

## JanKhan (main)

1. public class JanKhan {
2. static InstrumentUnit RocketMissionController;
3. /\*\* main Method \*/
4. public static void main(String[] args) {
5. RocketMissionController = new InstrumentUnit();
6. RocketMissionController.launchMission();
8. } // End of main method

11. }

## Propellent

1. public class Propellent {
2. private int weight;
3. public String type;
5. public Propellent(int quantity, String type) {
6. if (quantity >= 0) {
7. this.weight = quantity;
8. }
10. }
12. public int getWeight() {
13. return weight;
15. }
17. public void setWeight(int weight) {
18. this.weight = weight;
19. System.out.println("fuel set to " + weight + "kg");
20. }

23. }

## Rocket

1. public class Rocket {
2. Stage[] stages = new Stage[3];
3. Subject lunarModule;
4. String name;
6. Rocket(String name) {
7. stages[0] = new Stage(130000, 2290000 - 130000, ("Rocketdyne F-1"), 8400, 5, 165, 951000000);
8. stages[1] = new Stage(40100, 496200 - 40100, "Rocketdyne J-2", 1400, 5, 360, 5141000);
9. stages[2] = new Stage(13500, 123000 - 13500, "Rocketdyne J-2", 1400, 1, (165 + 335), 1000000);
10. lunarModule = new Subject(16000, 384400 \* 1000);
11. this.name = name;
12. }



17. public Engine getConnectedEngine() {
18. for (int i = 0; i < 3; i++) {
19. if (stages[i].mEngines.isRunning()) {
20. return stages[i].mEngines;
21. }
22. }
23. for (int i = 0; i < 3; i++) {
24. if (stages[i] != null)
25. return stages[i].mEngines;
26. }
27. return null;
29. }



34. public int getWeight() {
35. int temp = 0;
36. for (int i = 0; i < 3; i++) {
37. if (stages[i] != null) {
38. temp += stages[i].getGrossWeight();
39. }
40. }
41. return temp + lunarModule.weight;
42. }
44. public void startEngineOFStage(int stage) {
45. stages[stage - 1].mEngines.startEngine();
46. System.out.println("Engine of stage " + stage + " started.");
47. }
48. public void cutOffStage(int stage) {
49. stages[stage - 1] = null;
50. }

53. }

## Stage

1. public class Stage {
2. int numberOfEngines;
3. private int bodyWeight;
4. private int burnTime;
5. int thrust;
6. public Engine mEngines;
7. FuelTank mFuelTank;
9. public Stage(int bodyWeight, int fuelCapacity, String engineName, int engineWeight, int numberOfEngines, int burnTime, int thrust) {
10. this.burnTime = burnTime;
11. this.numberOfEngines = numberOfEngines;
12. this.bodyWeight = bodyWeight - numberOfEngines \* engineWeight;
13. this.mFuelTank = new FuelTank(fuelCapacity);
14. mEngines = new Engine(engineName, numberOfEngines \* engineWeight, this.mFuelTank, fuelCapacity / burnTime, thrust);
15. this.thrust = thrust;

18. }
19. public int getWeightOfEngines() {
20. return mEngines.weight;
21. }
23. public int getGrossWeight() {
24. return mFuelTank.getWeight() + bodyWeight + this.getWeightOfEngines();
25. }
27. public boolean isWorking() {
28. if (mFuelTank != null && mEngines != null) {
29. return true;
30. }
31. return false;
32. }


36. }

## Subject

1. public class Subject {
2. int weight;
3. long destinationDistance;
5. public Subject(int weight, long destinationDistance) {
6. this.weight = weight;
7. this.destinationDistance = destinationDistance;
8. }
10. }

## TimeScale:

1. import java.util.Date;


5. public abstract class TimeScale {
6. Date time;
7. long startTime;
8. long currentTime;
9. int interval = 1000;
11. public TimeScale() {


15. }
16. public void setInterval(int interval) {
17. this.interval = interval;
18. }
19. public void onSubclassClasObjectConstructed() {
20. time = new Date();
21. startTime = time.getTime();
22. while (true) {
23. currentTime = time.getTime();
24. if ((currentTime - startTime) % interval == 0) {
25. onIntervalChange();
26. }
28. }
29. }
31. public int getTotalSecondsPassed() {
32. return (int)(currentTime - startTime) / 1000;
33. }
35. abstract void onIntervalChange();





42. }