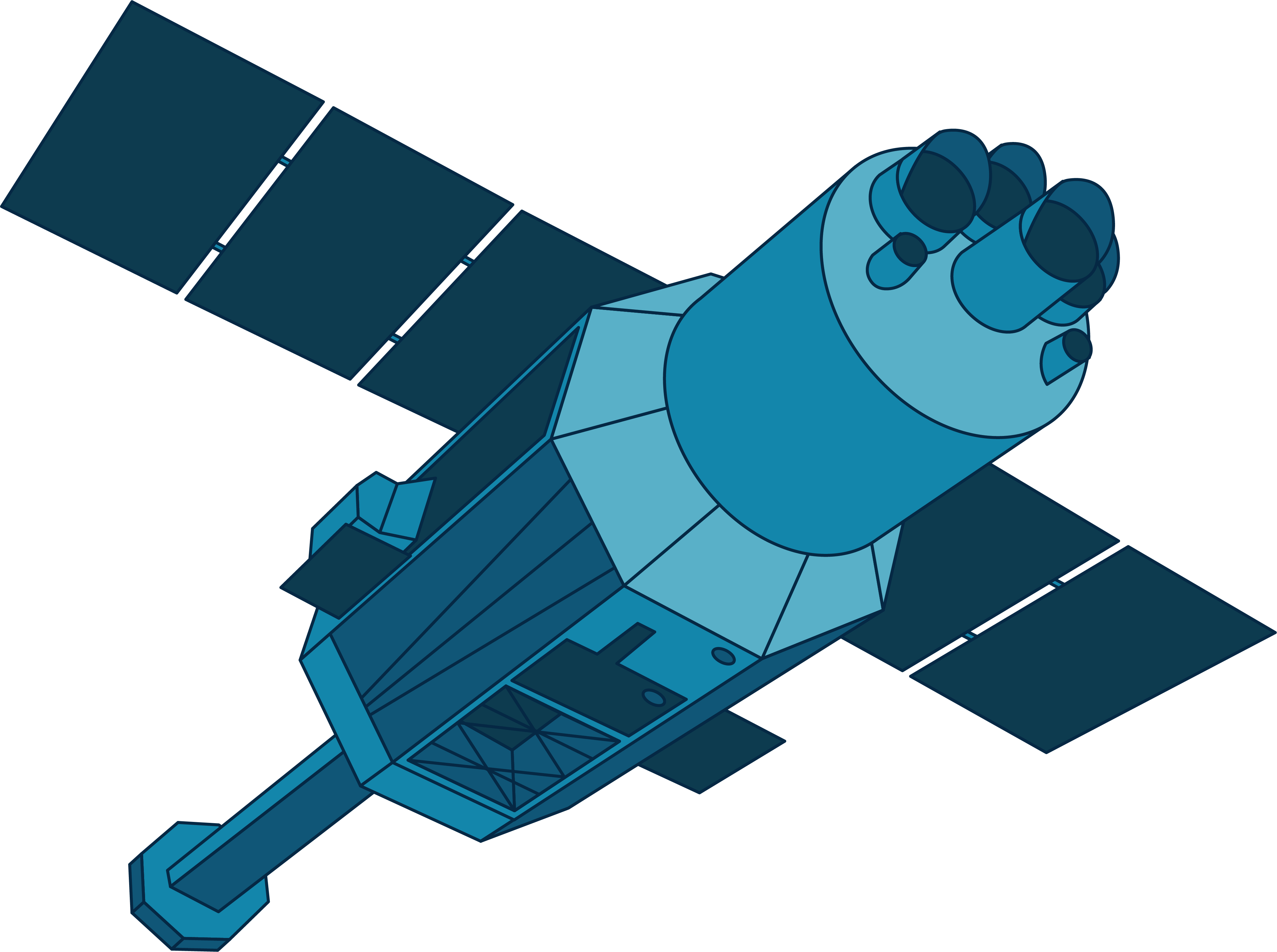
# 03. Spacecraft



*Spacecraft are vehicles designed to travel and operate in outer space. They serve*

## Preparation

Download the skeleton provided in Judge. **Do not** change the **packages**!

**Pay attention to name the package spaceCrafts, all the classes, their fields, and methods the same way they are presented in the following document. It is also important to keep the project structure as described.**

## Problem description

Your task is to create a repository that stores spacecraft by creating the classes described below.

### Spacecraft

First, write a class **Spacecraft** with the following properties:

* **name: String**
* **missionType: String**
* **target: String**
* **objective: String**
* **weight: int**

The class **constructor** should receive **name, missionType, target, objective** and **weight**. You need to create the appropriate **getters and setters**. All spacecraft names will be **unique.** It is guaranteed that there **will be no duplicates** of names.

Override the **toString()** method in the following format:  
**The mission of {name} was to reach {target} and to {objective}**

### LaunchPad

**Next**, write a class **LaunchPad**. The **LaunchPad** class should have those **properties**:

* **name: String**
* **capacity: int**
* **spacecrafts: List<Spacecraft>**

The class **constructor** should receive **name** and **capacity.** Also, it should initialise the **spacecrafts** with a new **collection** instance.Implement the following features:

* **Method addSpacecraft(Spacecraft spacecraft)** – **adds** an **entity** to the data **if** a spacefor it, otherwise print: **"This launchpad is at full capacity!"**
* **Method removeSpacecraft(String name)** – removes a **Spacecraft** by **given name,** if such **exists**, and **returns boolean** (true if it is removed, otherwise – false)
* **Method getHeaviestSpacecraft()**– **returns String** the **heaviest spacecraft** by **weight** in the given launchpad in the following format:
* **"{spaceCraft name} - {spaceCraft weight}kg."**
* **Method getSpacecraft(String name)** – **returns** the **spacecraft** with the **given name,** otherwise – returns **null**
* **Method getCount()** – **returns** the **count** of **spacecrafts** in the given launchpad
* **Method getSpacecraftsByMissionType(String missionType)** **–** **returns** **List** **–** a **collection** of **Spacecraft** whichholds the spacecrafts having the **same** **missionType** in the given launchpad.

In case there are **no** spacecrafts to **respond** to this **condition** (missionType) print:

* **"There are no spacecrafts to respond this criteria."**
* **Method getStatistics()** – **returns** a **String** in the following **format** (print the spacecrafts in **order of addition**):
  + **"Spacecrafts launched from {launchpad name}:  
    1. {spacecraft name}  
    2. {spacecraft name}  
     (…)**

1. **{spacecraft name}"**

If there are no spacecrafts in some launchpad print:

* **"Spacecrafts launched from {launchpad name}:**

**none"**

## Constraints

* The **name** and **weight** of the spacecraft will always be **unique**.
* You will always have a spacecraft added before receiving methods manipulating the Launchpad's spacecraft.

## Examples

This is an example of how the **Launchpad** class is **intended to be used**.

|  |
| --- |
| **Sample code usage** |
| //Initialize the repositories (LaunchPad)  LaunchPad kennedySC = **new** LaunchPad("Kennedy Space Center", 3);  LaunchPad capeCanaveralAFS = **new** LaunchPad("Cape Canaveral Air Force Station", 5);  //Initialize entities (Spacecraft)  Spacecraft galileo = **new** Spacecraft("Galileo", "Orbiter", "Europa", "study Jupiter and its mysterious moons", 223);  Spacecraft cassini = **new** Spacecraft("Cassini-Huygens", "Orbiter", "Saturn", "analyze the composition and atmosphere of Saturn", 2523);  Spacecraft magellan = **new** Spacecraft("Magellan", "Orbiter", "Venus", "image the entire surface of Venus", 3445);  Spacecraft huygens = **new** Spacecraft("Huygens", "Atmospheric Probe/Lander", "Titan", "provide a detailed study of Titan's atmosphere", 318);  Spacecraft voyager1 = **new** Spacecraft("Voyager-1", "Multiple Flybys", "Outer Solar System", "chart the edge of interstellar space", 733);  //Add Spacecraft  kennedySC.addSpacecraft(galileo);  kennedySC.addSpacecraft(cassini);  kennedySC.addSpacecraft(magellan);  kennedySC.addSpacecraft(huygens); //This launchpad is at full capacity!  capeCanaveralAFS.addSpacecraft(voyager1);  //Get Spacecraft By Mission Type  kennedySC.getSpacecraftsByMissionType("Orbiter").forEach(spacecraft -> System.***out***.println(spacecraft.getName()));  //Galileo  //Cassini-Huygens  //Magellan  capeCanaveralAFS.getSpacecraftsByMissionType("Multiple Flybys").forEach(spacecraft -> System.***out***.println(spacecraft.getName()));  //Voyager-1  //Remove Spacecraft  System.***out***.println(capeCanaveralAFS.removeSpacecraft("Voyager-1")); //true  //Launchpad Get Count Of Spacecrafts  System.***out***.println(capeCanaveralAFS.getCount()); //0  //Spacecraft toString()  System.***out***.println(galileo);  //The mission of Galileo was to reach Europa and to study Jupiter and its mysterious moons  //Get Heaviest Spacecraft  System.***out***.println(kennedySC.getHeaviestSpacecraft()); //Magellan - 3445kg.  //LaunchPad Get Statistics  System.***out***.println(kennedySC.getStatistics());  //Spacecrafts launched from Kennedy Space Center:  //1. Galileo  //2. Cassini-Huygens  //3. Magellan  System.***out***.println(capeCanaveralAFS.getStatistics());  //Spacecrafts launched from Cape Canaveral Air Force Station:  //none |

## Submission

Submit **single .zip file**, containing **spaceCrafts** package, **with the classes inside** (**Launchpad**, **Spacecraft** and the **Main** **class)**, there is no specific content required inside the **Main** class e.g. you can do any kind of local testing of your program there. However, there should be **main(String[] args)** method inside.