The Government of India embarked on an ambitious project to conduct a comprehensive landscape survey of the country. To achieve this, they approached the aeronautical engineering departments of top institutes, seeking their expertise in drone technology and to handle critical checks and calculations using exception handling.

Your task is to design a Java program to receive and validate drone survey data (batteryLevel, gpsSignal, droneType, weightCapacity) and calculate flight time based on droneType,speed, and distance. Incorporate exception handling to manage various error scenarios associated with these inputs.

**Component Specification: DroneManagementSystem (Utility class)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type (Class)** | **Method** | **Parameters** | **Responsibilities** |
| DroneManagementSystem | checkBatteryLevel | double batteryLevel | This method accepts batteryLevel as an input parameter and verifies the current battery level of a drone to ensure it is sufficient for flight operations. If the battery level is below 20%, it throws an **InvalidDroneDataException**with an appropriate message **"Battery level is too low for the flight: <batteryLevel> %",**otherwise, it returns true. |
| DroneManagementSystem | checkGpsSignal | double gpsSignalStrength | This method accepts gpsSignalStrength as an input parameter and verifies the current GPS signal strength of a drone to ensure it is sufficient for flight operations. If the signal strength is below 50%, it throws an **InvalidDroneDataException**with a message; **"GPS signal strength is too weak for the flight: <gpsSignalStrength> %"**, otherwise, it returns true. |
| DroneManagementSystem | checkDroneType | String droneType | This method verifies if the provided droneType is valid. It accepts a droneType as an input parameter and the system supports 2 droneType : "**Quadcopter**" or "**Fixed-wing"**. If it matches either of these types, the method returns true. If not, it throws an **InvalidDroneDataException**with a message **"<droneType> is an invalid Drone Type.".**  **Constraint:**   * The droneType is case-sensitive. |
| DroneManagementSystem | checkWeightCapacity | String droneType, double weight | This method accepts droneType and weight as input parameters and verifies that the weight being carried by the drone does not exceed its maximum capacity.The system supports 2 droneType : Quadcopter and Fixed-wing.For a **"Quadcopter"** drone, the maximum weight capacity is 10 kg, and for a **"Fixed-wing"** drone, the maximum weight capacity is 5 kg. If the weight exceeds these limits, the method throws an **InvalidDroneDataException**with a message **"Weight exceeds the maximum capacity for <droneType>: <weight> kg".** |
| DroneManagementSystem | calculateFlightTime | double speed, double distance | This method accepts speed, and distance as input parameters and calculates the flight time of a drone based on speed over a given distance. The method calculates the flight time using the formula:  Flight Time=Distance/Speed  The calculated flight time is then returned as double. |

**Component Specification:**

**InvalidDroneDataException (This class inherits the Exception Class)**

| **Type (Class)** | **Responsibilities** |
| --- | --- |
| **InvalidDroneDataException** | The single-argument constructor public **InvalidDroneDataException**(String message) is used to throw an exception when **batteryLevel, gpsSignalStrength, weightCapacity, or droneType**  do not follow the validation rules. |

If any exception is raised during validation, handle it appropriately and display an error message corresponding to the type of exception. 

The main method in the UserInterface class starts the program, prompting the user to input drone details**(ID:type:batteryLevel:gpsSignal:weight:speed)**. It parses these details, creates instances of **DroneManagementSystem**and Drone, and validates the drone's battery level, GPS signal strength, and weight capacity. The method then asks for the distance the drone should cover, calculates the flight time, and prints the result. If any validation fails, it catches and handles the **InvalidDroneDataException**, displaying an error message.

The output should be printed as shown in the sample outputs.

**Note:**

* **Propagate the exceptions that occur in InvalidDroneDataException class and handle them in main method.**
* If the input results in **InvalidDroneDataException**then print the corresponding message.
* In the sample input and output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.
* Ensure to follow the object-oriented specifications provided in the question description.
* Ensure to provide the names for classes, attributes, and methods as specified in the question description.
* Adhere to the code template, if provided.
* Do not use **System.exit(x)** to terminate the code.

**Sample Input/Output 1:**

Enter the drone details:

**D123:Quadcopter:15:75:8:50**

Battery level is too low for the flight: 15.0%

**Sample Input/Output 2:**

Enter the drone details:

**D124:Fixed-wing:80:45:4:60**

GPS signal strength is too weak for the flight: 45.0%

**Sample Input/Output 3:**

Enter the drone details:

**D125:Quadcopter:85:80:12:40**

Weight exceeds the maximum capacity for Quadcopter: 12.0kg

**Sample Input/Output 4:**

Enter the drone details:

**D125:Fixed-wing:85:80:10:40**

Weight exceeds the maximum capacity for Fixed-wing: 10.0kg

**Sample Input/Output 5:**

Enter the drone details:

**D127:Hexacopter:95:90:3:55**

Hexacopter is an invalid Drone Type.

**Sample Input/Output 6:**

Enter the drone details:

**D128:Quadcopter:85:75:8:50**

Enter the distance drone should cover

**100**

D128 (Quadcopter) can fly for 2.0 hours to cover the distance of 100.0 km.

**Explanation**: In this scenario, the batteryLevel (85%) is above the 20% threshold, the gpsSignalStrength (75%) is above the 50% threshold, and the weight (8 kg) is within the 10 kg limit for a Quadcopter. The flight time is calculated based on the speed (50 km/h) and the distance (100 km), resulting in a flight time of 2 hours.

**Sample Input/Output 7:**

Enter the drone details:

**D129:Fixed-wing:90:80:4:60**

Enter the distance drone should cover

**120**

D129 (Fixed-wing) can fly for 2.0 hours to cover the distance of 120.0 km.

**Explanation**: In this scenario, the batteryLevel (90%) is above the 20% threshold, the gpsSignalStrength (80%) is above the 50% threshold, and the weight (4 kg) is within the 5 kg limit for a Fixed-wing drone. The flight time is calculated based on the speed (60 km/h) and the distance (120 km), resulting in a flight time of 2 hours.