```
public class Trampala {

// This Trampala is represented by it's MaxLoad

private K(maxLoad), lWeight, rWeight, totalWeight;

///////////////CustomExceptions///////////////
// Exception: ItemNotInListException
// Precondition:
// Postcondition:
// Throws it when use remove methods and
// item you want to remove is not contained at Trampala
class ItemNotInListException extends Exception;

///////////////Constructors///////////////
// Constructor: Trampala
// Invariants: MaxLoad is a positive integer
// Precondition:
// a) maxLoad > 0
// Postcondition:
// Constructs a valid Trampala with positive maxLoad
public Trampala(int MaxLoad);

///////////////Accesors///////////////
// Method: getMaxLoad
// Invariants: Trampala is constructed and not broken
// Precondition: -
// Postcondition:
// Returns the MaxLoad of the current Trampala
public void getMaxLoad();
// Method: getTotalWeight
// Invariants: Trampala is not broken and TotalWeight Set
// Precondition: -
// Postcondition:
// Returns the totalWeight that has added to the current Trampala
public void getTotalWeight();

///////////////Observers///////////////
// Method: contains
// Invariants: Trampala is not broken
// Precondition:
// a) w is a valid instance of item
// Postcondition:
// Returns true if Trampala contains w, otherwise false
public boolean contains(Item w);
// Method: isBroken
// Invariants: Trampala is constructed
// Precondition: -
// Postcondition:
// Returns true if Trampala is broken, otherwise false
public boolean isBroken();
// Method: isBalanced
```

```
// Invariants: Trampala is not broken
// Precondition: -
// Postcondition:
// Returns true if Trampala is balanced, otherwise false
public boolean isBalanced();

///////////////Transformers///////////////
// Method: setMaxLoad
// Invariants: MaxLoad is a positive integer
// Precondition:
// a)MaxLoad > 0
// Postcondition:
// Sets the MaxLoad of the current Trampala
public void setMaxLoad(int k);
// Method: addLeft
// Invariants: Trampala is not broken and Item w is not null
// Precondition:
// a)w is a valid instance of Item
// Postcondition:
// Adds the item to the left side of the Trampala
public void addLeft(Item w);
// Method: addRight
// Invariants: Trampala is not broken and Item w is not null
// Precondition:
// a)w is a valid instance of Item
// Postcondition:
// Adds the item to the right side of the Trampala
public void addRight(Item w);
// Method: removeLeft
// Invariants: Trampala is not broken and Item w is not null
// Precondition:
// a)left side of Trampala already contains w
// Postcondition:
// Removes the item from the left side of the Trampala
public void removeLeft(Item w);
// Method: removeRight
// Invariants: Trampala is not broken and Item w is not null
// Precondition:
// a)right side of Trampala already contains w
// Postcondition:
// Removes the item from the right side of the Trampala
public void removeRight(Item w);

}




public class Item{
///////////////Constructors///////////////
// Constructor: Item
```

```
// Invariants: Name is a valid instance of String / weight integer > 0
// Precondition:
// a) name is a valid instance of char
// b) weight is a valid instance of int
// Postcondition:
// Constructs a valid Item with name and weight
public Item(char name);

//////////////Accesors///////////////
// Method: getName
// Invariants: Name field is not null
// Precondition: -
// Postcondition:
// Returns the name of the current Item
public void getName();
// Method: getWeight
// Invariants: Weight field is not null
// Precondition: -
// Postcondition:
// Returns the weight of the current Item
public void getWeight();

//////////////Transformers///////////////
// Method: setName
// Invariants: Name is valid instance of string
// Precondition:
// a)name is a valid instance of string
// Postcondition:
// Sets the given name to the Item
public void setName(string name);
// Method: setWeight
// Invariants: Weight must be an integer > 0
// Precondition:
// a)Weight must be > 0
// Postcondition:
// Sets the given weight to the Item
public void setWeight(int weight);

}
```