Assignment 1

CS 1083 FR01A

Kisenge Mbaga

Container Class

```
/**
This class represents a Container.

@author Kisenge Mbaga
*/

public class Container{


    private int containerWeight;
    private boolean hazardous;
    private String portName;
    private static int containerId = 1000;




    /**
    This method constructs a Ship.
    @param portNameIn the port city of departure.
    @param containerWeightIn the containers weight in kg.
    @param hazardousIn the maximun weight allowed on the ship .
    */

    public Container (String portNameIn, int containerWeightIn , boolean hazardousIn){

      portName = portNameIn;
      containerWeight = containerWeightIn;
      hazardous = hazardousIn;
      containerId++;

    }



    /**
    This method returns the Id.
    @return the container's Id.
    */
    public int getId(){
      return containerId;
    }

    /**
    This method returns the port city of departure.
    @return the port city of departure.
    */
    public String getPort(){
      return portName;
    }
```

```
    /**
    This method returns the container's weight.
    @return the container's weight.
    */
    public int getWeight(){
      return containerWeight;
    }


    /**
    This method returns whether a container is hazardous.
    @return whether a container is hazardous.
    */
    public boolean getHazardous(){
      return hazardous;
    }


    /**
    This method changes a container's weight.
    @param newWeight the new weight of the container .
    */
    public int setWeight(int newWeight){
      containerWeight = newWeight;
      return newWeight;
    }



}
```

Ship Class

```
/**
This class represents a Container.

@author Kisenge Mbaga
*/


public class Ship {
```

```
   private int maxHazardous;
   private int maxTotalContainter;
   private int maxWeight;
   private Container[] containerList;
   private int containerNumCurrent;
   private int weightCurrent=0;
   private int hazardousCurrent;


   /**
   This method constructs a Ship
   @param maxHazardousIn the max number of hazardous containers allowed.
   @param maxTotalIn the max number of containers allowed.
   @param maxWeightIn the maximun weight allowed on the ship .
   */

   public Ship(int maxHazardousIn, int maxTotalIn, int maxWeightIn) {
      maxHazardous = maxHazardousIn;
      maxTotalContainter = maxTotalIn;
      maxWeight = maxWeightIn;
      containerList = new Container[maxTotalContainter];

   }


   /**
   This method adds a container to the Ship.
   @param containerNew the container being added
   @return whether the container was successfully added.
   */

   public boolean addContainer(Container containerNew) {
      boolean temp=false;
      Container [] containerTemp = new Container[maxTotalContainter];
      int hazardousTemp=0;


      if (containerNumCurrent < maxTotalContainter) { //if not full

         for (int i = 0; i < containerNumCurrent+1; i++) { //if the container is Hazardous, checking tht
doesnt put over Hazard limit
            if (containerNew.getHazardous()==true){
               hazardousTemp=1;
            }
            if (weightCurrent + containerNew.getWeight() < maxWeight &&
            (hazardousCurrent + hazardousTemp) <= maxHazardous)  {//if adding container doesnt put
ship over weight and hazard limit
               temp = true;
            }
            else {
```

```
                temp = false;
              }
            }

          }
        else {
           temp = false;
        }


        if (temp==true){
           for (int i= containerNumCurrent ; i< containerNumCurrent+1; i++){// add container to temp array,
and add to real array if not already on ship
              containerTemp[i] = new Container(containerNew.getPort(), containerNew.getWeight(),
containerNew.getHazardous());
              Container temp2 = containerTemp[i];

              if ( temp2.equals(containerNew)){
                 System.out.print("Boat is already on board.");
              }

              else {
                 containerList[i] = new Container(containerNew.getPort(), containerNew.getWeight(),
containerNew.getHazardous());
                 System.out.print("This container is now listed. The real container ID is "+
containerList[i].getId()+ "\n");
              }
     }

     //update values
     if (containerNew.getHazardous()){ //Hazard checker.
        hazardousCurrent++;
     }
     containerNumCurrent++;
     weightCurrent += containerNew.getWeight();
     return true;
   }


   else {
     System.out.print("Cant add\n");
     //temp=true;
     return false;
     }


 }


   /**
   This method adds a container to the Ship.
```

```
    @param containerRemove the container being removed
    @return whether the container was successfully removed.
    */
    public boolean removeContainer(Container containerRemove) {
       boolean temp;
       int removeCounter=0;
       Container [] containerTemp = new Container[maxTotalContainter];

       for (int i = 0; i < containerNumCurrent; i++) {
         containerTemp[i] = new Container(containerRemove.getPort(), containerRemove.getWeight(),
containerRemove.getHazardous());

         Container temp2 = containerTemp[i];


           if (containerRemove.getPort().equals(temp2.getPort())) {

             removeCounter=i;


               temp = true;
           }
       }


       if (temp=true){
         for (int i=removeCounter; i < containerNumCurrent-1; i++){
           containerList[i]=containerList[i+1];
         }

         if (containerRemove.getHazardous()) { //Hazard checker.
            hazardousCurrent--;
         }

         containerNumCurrent--;
         weightCurrent -= containerRemove.getWeight();

         return true;
       }

       else {return false;}
    }


    /**
    This method adds a container to the Ship.
    @param containerUpdate the container being removed

    */

    public void updateWeight(Container containerUpdate, int weightNew) {
```

```
  Container [] containerTemp= new Container[maxTotalContainter];

    for (int i = 0; i < containerNumCurrent; i++) {

      Container temp2 = containerList[i];


        if (containerUpdate.equals(temp2)  && ((weightCurrent + weightNew) < maxWeight)) {
           temp2.setWeight(weightNew);
        } else {

        }
    }
}



/**
This method displays a list of the containers on the ship.

*/
public void displayData() {
  int weightTotal=0;
    for (int i = 0; i < containerNumCurrent; i++) {

        System.out.print("\n"+containerList[i].getId() + "\t");
        System.out.print(containerList[i].getPort() + "\t");
        System.out.print(containerList[i].getWeight() + ".00kg" + "\t");
        weightTotal += containerList[i].getWeight();
        System.out.print(containerList[i].getHazardous() + "\n");
    }
      System.out.print("Total Weight: " + weightTotal+ ".00kg\n");

}



/**
This method gets the available weight on a ship.
@return the available weight

*/
public int availableWeight() {
    int weightTotal=0;
    Container [] containerTemp= new Container[maxTotalContainter];

    for (int i = 0; i < containerNumCurrent; i++) {
      Container temp2 = containerList[i];
      weightTotal = weightTotal + temp2.getWeight();
    }
```

```
      System.out.print("Available space:" + (maxWeight-weightTotal));
      return maxWeight - weightTotal;

   }


}
```

ShipDriver

```
/**
This class is the driver for the Ship class.

@author Kisenge Mbaga
*/

public class ShipDriver{

    public static void main(String[] args){


    Ship unsinkable = new Ship (2,5,10000);



    Container blue = new Container ("London",2000,true);
      Container red = new Container ("Cairo",1200, false);
    Container green = new Container ("Halifax", 3000, false);
    Container orange = new Container ("Vancouver", 1000, true);

    Container overWeight = new Container ("San Francisco", 6000, false);
    Container tooHazardous = new Container ("Shanghai", 2000, true);



    unsinkable.addContainer(red);
    unsinkable.addContainer(blue);
    unsinkable.addContainer(green);
    unsinkable.addContainer(orange);

    unsinkable.addContainer(overWeight);
    unsinkable.addContainer(tooHazardous);


    unsinkable.updateWeight(green,2500);
```

```
unsinkable.removeContainer(orange);

unsinkable.availableWeight();

unsinkable.displayData();

}
}
```

Output

```
C:\Users\kisen\Java2\Assignments\Assignment_1>java ShipDriver
This container is now listed.1008
This container is now listed.1010
This container is now listed.1012
This container is now listed.1014
Cant add
Cant add
Available space:4300
1018    Cairo   1200.00kg       false

1018    London  2000.00kg       true

1018    Halifax 2500.00kg       false
Total Weight: 5700.00kg
```

Red, blue, green and orange were added successfully.

The ships that could not be added would put the ship overweight and over the hazardous limit. The available space was 4300, while Overweight was 6000.

Orange (Vancouver) was removed successfully.

Halifax's weight was changed successfully from 3000 to 2500.

For some reason I could not print the correct Ids in the list, but I have printed them above.