

# GarbageSortingSystem

December 10, 2021

## Contents

|           |                                 |           |
|-----------|---------------------------------|-----------|
| <b>1</b>  | <b>AddressRepository</b>        | <b>1</b>  |
| <b>2</b>  | <b>GarbagePack</b>              | <b>2</b>  |
| <b>3</b>  | <b>GarbageType</b>              | <b>3</b>  |
| <b>4</b>  | <b>Glass</b>                    | <b>4</b>  |
| <b>5</b>  | <b>Metal</b>                    | <b>4</b>  |
| <b>6</b>  | <b>Paper</b>                    | <b>5</b>  |
| <b>7</b>  | <b>Plastic</b>                  | <b>5</b>  |
| <b>8</b>  | <b>GarbageSorter</b>            | <b>6</b>  |
| <b>9</b>  | <b>GarbageSortingController</b> | <b>6</b>  |
| <b>10</b> | <b>GarbageSortingSystem</b>     | <b>8</b>  |
| <b>11</b> | <b>GarbageTruck</b>             | <b>9</b>  |
| <b>12</b> | <b>RecyclingPlant</b>           | <b>11</b> |
| <b>13</b> | <b>GarbageSortingTest</b>       | <b>12</b> |
| <b>14</b> | <b>GarbageTruckTest</b>         | <b>14</b> |
| <b>15</b> | <b>TRunner</b>                  | <b>15</b> |
| <b>16</b> | <b>Test</b>                     | <b>16</b> |
| <b>17</b> | <b>TestCase</b>                 | <b>16</b> |
| <b>18</b> | <b>TestResult</b>               | <b>17</b> |
| <b>19</b> | <b>TestSuite</b>                | <b>18</b> |
| <b>20</b> | <b>Environment</b>              | <b>18</b> |
| <b>21</b> | <b>GLOBAL</b>                   | <b>20</b> |

# 1 AddressRepository

```

class AddressRepository is subclass of GLOBAL

instance variables
  addresses_ : map seq of char to set of GarbagePack := {|->};
  inv InvAddressID(addresses_, InvalidAddressChars);

functions

private InvAddressID : map seq of char to set of GarbagePack* set of char -> bool
InvAddressID(addr, invalidChars) ==
  forall adr in set dom addr &
    forall s in seq adr &
      forall ch in set invalidChars & s <> ch;

operations

public AddressRepository : map seq of char to set of GarbagePack ==> AddressRepository
AddressRepository(aMap) ==
(
  addresses_ := aMap;
);

public addToAddresses : map seq of char to set of GarbagePack ==> ()
addToAddresses(aMap) ==
(
  for all addr in set dom aMap do
  (
    addresses_(addr) := addresses_(addr) union aMap(addr)
  )
)
pre forall aP in set dom aMap &
  exists1 p in set dom addresses_ & aP = p

-- Dunion can be used because objects are used and therefore parts of the set wont be removed if
  they are the same.
post forall gR in set dunion rng aMap &
  exists1 p in set dunion rng addresses_ & p = gR;

pure public getGarbageFromAddress : seq of char ==> set of GarbagePack
getGarbageFromAddress(addr) ==
(
  return addresses_(addr)
)
pre exists1 p in set dom addresses_ & p = addr;

public removeGarbageFromAddress : seq of char * GarbagePack ==> ()
removeGarbageFromAddress(addr, GarbagePack) ==
(
  addresses_(addr) := addresses_(addr) \ {GarbagePack};

```

```

)
pre exists1 p in set dom addresses_ & p = addr
post forall p in set rng addresses_ &
    forall gp in set p & GarbagePack <> gp;

pure public getAddresses : () ==> map seq of char to set of GarbagePack
getAddresses() ==
(
    return addresses_;
);

end AddressRepository

```

| Function or operation    | Line | Coverage | Calls |
|--------------------------|------|----------|-------|
| AddressRepository        | 17   | 100.0%   | 7     |
| InvAddressID             | 9    | 100.0%   | 3185  |
| addToAddresses           | 23   | 100.0%   | 45    |
| getAddresses             | 56   | 100.0%   | 8     |
| getGarbageFromAddress    | 39   | 100.0%   | 120   |
| removeGarbageFromAddress | 46   | 100.0%   | 60    |
| AddressRepository.vdmpp  |      | 100.0%   | 3425  |

## 2 GarbagePack

```

class GarbagePack is subclass of GLOBAL

instance variables

garbageSet : set of GarbageType := {};
inv card garbageSet <= 10;

operations

public GarbagePack : set of GarbageType ==> GarbagePack
GarbagePack(garbage) ==
(
    garbageSet := garbage;
);

pure public getGarbagePack : () ==> set of GarbageType
getGarbagePack() ==
(
    return garbageSet;
);

pure public getPackWeight : () ==> nat
getPackWeight() ==
(
    return SumWeightGarbagePack(getGarbagePack());
);

```

```

pure public getPackVolume : () ==> nat
getPackVolume() ==
  return SumVolumeGarbagePack (getGarbagePack ())
end GarbagePack

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| GarbagePack           | 9    | 100.0%   | 87    |
| getGarbagePack        | 15   | 100.0%   | 1227  |
| getPackVolume         | 27   | 100.0%   | 242   |
| getPackWeight         | 21   | 100.0%   | 484   |
| GarbagePack.vdmpp     |      | 100.0%   | 2040  |

### 3 GarbageType

```

class GarbageType is subclass of GLOBAL

instance variables
public weight: nat := 0;
inv weight > 0 and weight < GARBAGETYPE_MAX_WEIGHT;
protected garbageId : [GLOBAL'GarbageId] := nil;

public dimensions: [GLOBAL'dimensionsType] := nil;
inv InvDimensions(dimensions)

functions

private InvDimensions : GLOBAL'dimensionsType -> bool
InvDimensions(mk_GLOBAL'dimensionsType(width,length,height)) ==
  width < GARBAGETYPE_MAX_WIDTH and width > 0 and
  length < GARBAGETYPE_MAX_LENGTH and length > 0 and
  height < GARBAGETYPE_MAX_HEIGHT and height > 0;

operations

pure public getWeight : () ==> nat
getWeight() ==
  return weight;

pure public getVolume : () ==> nat
getVolume() ==
  return dimensions.width*dimensions.length*dimensions.height;

pure public getGarbageId : () ==> GLOBAL'GarbageId
getGarbageId() ==
  return garbageId;

end GarbageType

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| InvDimensions         | 14   | 100.0%   | 972   |
| getGarbageId          | 31   | 100.0%   | 183   |
| getVolume             | 27   | 100.0%   | 1458  |
| getWeight             | 23   | 100.0%   | 2370  |
| GarbageType.vdmpp     |      | 100.0%   | 4983  |

## 4 Glass

```

class Glass is subclass of GarbageType

operations

public Glass : GLOBAL`dimensionsType * nat ==> Glass
Glass(d, w) ==
  atomic
  (
    dimensions := d;
    weight := w;
    garbageId := <GLASSID>
  );

end Glass

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| Glass                 | 4    | 100.0%   | 72    |
| Glass.vdmpp           |      | 100.0%   | 72    |

## 5 Metal

```

class Metal is subclass of GarbageType

operations

public Metal : GLOBAL`dimensionsType * nat ==> Metal
Metal(d, w) ==
  atomic
  (
    dimensions := d;
    weight := w;
    garbageId := <METALID>
  );

end Metal

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| Metal                 | 4    | 100.0%   | 98    |

|             |  |        |    |
|-------------|--|--------|----|
| Metal.vdmpp |  | 100.0% | 98 |
|-------------|--|--------|----|

## 6 Paper

```

class Paper is subclass of GarbageType

operations

public Paper : GLOBAL'dimensionsType * nat ==> Paper
Paper(d, w) ==
  atomic
  (
    dimensions := d;
    weight := w;
    garbageId := <PAPERID>
  );

end Paper

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| Paper                 | 4    | 100.0%   | 98    |
| Paper.vdmpp           |      | 100.0%   | 98    |

## 7 Plastic

```

class Plastic is subclass of GarbageType

operations

public Plastic : GLOBAL'dimensionsType * nat ==> Plastic
Plastic(d, w) ==
  atomic
  (
    dimensions := d;
    weight := w;
    garbageId := <PLASTICID>
  );

end Plastic

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| Plastic               | 4    | 100.0%   | 56    |
| Plastic.vdmpp         |      | 100.0%   | 56    |

## 8 GarbageSorter

```

class GarbageSorter is subclass of GLOBAL

functions
--(<GLASS> |-> {x})
--(<GLASS> |-> {y})
--(<GLASS> |->{y, x})
-- If domaine of m1, is in m2, then union and add too same dom
-- else add both m1 and m2 to map, with each different dom

public MapCombine : GarbageMap * GarbageMap -> GarbageMap
MapCombine(m1, m2) ==
(
    {id |-> m1(id) union m2(id) | id in set dom m1 inter dom m2} munion
    {id |-> m1(id) | id in set dom m1 \ dom m2} munion
    {id |-> m2(id) | id in set dom m2 \ dom m1}
);

public sortSetofGarbageType: set of GarbageType -> GarbageMap
sortSetofGarbageType(s) ==
    if s = {}
    then {}|->{}
    else
        let shead in set s
        in MapCombine({shead.getGarbageId()|->{shead}} , sortSetofGarbageType(s\{shead}))
measure card s;

end GarbageSorter

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| MapCombine            | 10   | 100.0%   | 90    |
| sortSetofGarbageType  | 18   | 100.0%   | 66    |
| GarbageSorter.vdmpp   |      | 100.0%   | 156   |

## 9 GarbageSortingController

```

class GarbageSortingController

instance variables
trucks : set of GarbageTruck := {};

operations

public Step : () ==> ()
Step() ==
(
    dcl fulltrucks : set of GarbageTruck;
    fillTrucks();
    fulltrucks := scanForFullTrucks();
    sendTrucksToPlant(fulltrucks);
);

```

```

public addTrucks : set of GarbageTruck ==> ()
addTrucks(gtset) ==
(
    trucks := trucks union gtset;
);

-- Will go through trucks, and then it will remove the trucks from the instance variable
-- set if they are full, and add them to a local variable that will be returned

private scanForFullTrucks : () ==> set of GarbageTruck
scanForFullTrucks() ==
(
    dcl fulltrucks : set of GarbageTruck := {t | t in set trucks & t.hasTruckBeenFilled()};
    trucks := trucks \ {t | t in set trucks & t.hasTruckBeenFilled()};
    return fulltrucks;
);

private fillTrucks : () ==> ()
fillTrucks() ==
(
    for all addrs in set dom GarbageSortingSystem`addressRepository.getAddresses()
    do
        (
            dcl gbFromAddr : set of GarbagePack := GarbageSortingSystem`addressRepository.
            getGarbageFromAddress(addrs);
            for all gbs in set gbFromAddr
            do
                (
                    for all t in set trucks
                    do
                        (
                            if ((not t.isTruckFull())
                                and (gbFromAddr <> {}))
                                and (not GarbageTruck`willBeOverfilled(t.getTruckGarbage(), gbs.
                                getPackWeight(), gbs.getPackVolume())) then
                                    (
                                        t.addToTruck(gbs);
                                        gbFromAddr := gbFromAddr \ {gbs};
                                        GarbageSortingSystem`addressRepository.removeGarbageFromAddress(
                                            addrs, gbs);
                                    )
                                else if (GarbageTruck`willBeOverfilled(t.getTruckGarbage(), gbs.
                                getPackWeight(), gbs.getPackVolume())) then
                                    (
                                        t.truckHasBeenFilled();
                                    )
                                )
                            );
                        )
                    );
                )
            );
        );
    );
);

private sendTrucksToPlant : set of GarbageTruck ==> ()
sendTrucksToPlant(truck) ==
(
    GarbageSortingSystem`plant.addFilledTrucksToPlant(truck)
);

end GarbageSortingController

```



| Function or operation          | Line | Coverage | Calls |
|--------------------------------|------|----------|-------|
| Step                           | 7    | 100.0%   | 20    |
| addTrucks                      | 16   | 100.0%   | 32    |
| fillTrucks                     | 32   | 100.0%   | 270   |
| scanForFullTrucks              | 24   | 100.0%   | 20    |
| sendTrucksToPlant              | 63   | 100.0%   | 20    |
| GarbageSortingController.vdmpp |      | 100.0%   | 362   |

## 10 GarbageSortingSystem

```

class GarbageSortingSystem

instance variables

public static trucks : set of GarbageTruck := {new GarbageTruck("ID1"), new GarbageTruck("ID2"),
    new GarbageTruck("ID3")};

public static garbageSortingController : GarbageSortingController := new GarbageSortingController
();

public static addressRepository : AddressRepository := new AddressRepository({ "Brammersgade"
    |-> {},
                                           "Frederiksgade"
    |-> {},
                                           "Odensegade"
    |-> {}});

public static plant : RecyclingPlant := new RecyclingPlant()

end GarbageSortingSystem

```

| Function or operation      | Line | Coverage | Calls |
|----------------------------|------|----------|-------|
| GarbageSortingSystem.vdmpp |      | 100.0%   | 0     |

## 11 GarbageTruck

```

class GarbageTruck is subclass of GLOBAL

instance variables

garbageTruckId_ : [seq of char] := nil;
inv (garbageTruckId_ = nil) or INVtruckId(garbageTruckId_, allowedIdNbrs);

hasBeenFilled : bool := false;

garbagePackSet_ : set of GarbagePack := {};
inv not (checkTruckWeight(garbagePackSet_)) and not (checkTruckDimension(garbagePackSet_))

functions

```

```

private INVtruckId : seq of char * set of char -> bool
INVtruckId(id, allowedIDNbrs) ==
(
    forall str_i in set {3, ..., len id} &
        exists p in set allowedIDNbrs & id(str_i) = p
    and id(1) = 'I' and id(2) = 'D'
);

private checkTruckWeight : set of GarbagePack -> bool
checkTruckWeight(gpset) ==
(
    GLOBAL'SumSet({SumWeightGarbagePack(i.getGarbagePack()) | i in set gpset & gpset <> {}}) >=
        GARBAGETRUCK_MAX_WEIGHT
);

private checkTruckDimension : set of GarbagePack -> bool
checkTruckDimension(gpset) ==
(
    GLOBAL'SumSet({SumVolumeGarbagePack(i.getGarbagePack()) | i in set gpset & gpset <> {}}) >=
        GARBAGETRUCK_MAX_VOLUME -- = 10 Max Sizes of GarbageTypes
);

public willBeOverfilled : set of GarbagePack * nat * nat -> bool
willBeOverfilled(gpset, w, vol) ==
(
    GLOBAL'SumSet({SumWeightGarbagePack(i.getGarbagePack()) | i in set gpset & gpset <> {}}) + w
        >= GARBAGETRUCK_MAX_WEIGHT
    or GLOBAL'SumSet({SumVolumeGarbagePack(i.getGarbagePack()) | i in set gpset & gpset <> {}}) +
        vol >= GARBAGETRUCK_MAX_VOLUME
);

operations

public GarbageTruck : seq of char ==> GarbageTruck
GarbageTruck(id) ==
(
    garbageTruckId_ := id;
)
pre INVtruckId(id, allowedIdNbrs)
post garbageTruckId_ <> nil;

public truckHasBeenFilled : () ==> ()
truckHasBeenFilled() ==
(
    hasBeenFilled := true;
);

pure public hasTruckBeenFilled : () ==> bool
hasTruckBeenFilled() ==
(
    return hasBeenFilled;
);

pure public isTruckFull : () ==> bool
isTruckFull() ==
(
    return checkTruckWeight(getTruckGarbage()) and checkTruckDimension(getTruckGarbage());

```

```

);

pure public getTruckGarbage : () ==> set of GarbagePack
getTruckGarbage() ==
(
    return garbagePackSet_;
);

public addToTruck : GarbagePack ==> ()
addToTruck(gp) ==
(
    garbagePackSet_ := garbagePackSet_ union {gp};
)
pre not GarbageTruck `willBeOverfilled(getTruckGarbage(), gp.getPackWeight(), gp.getPackVolume());

public emptyTruck : () ==> ()
emptyTruck() ==
(
    garbagePackSet_ := {};
    hasBeenFilled := false;
)

end GarbageTruck

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| GarbageTruck          | 43   | 100.0%   | 66    |
| INVtruckId            | 14   | 100.0%   | 133   |
| addToTruck            | 75   | 100.0%   | 122   |
| checkTruckDimension   | 28   | 100.0%   | 40    |
| checkTruckWeight      | 22   | 100.0%   | 484   |
| emptyTruck            | 83   | 100.0%   | 25    |
| getTruckGarbage       | 69   | 100.0%   | 420   |
| hasTruckBeenFilled    | 57   | 100.0%   | 120   |
| isTruckFull           | 63   | 71.4%    | 152   |
| truckHasBeenFilled    | 51   | 100.0%   | 35    |
| willBeOverfilled      | 35   | 100.0%   | 242   |
| GarbageTruck.vdmpp    |      | 98.4%    | 1839  |

## 12 RecyclingPlant

```

class RecyclingPlant

instance variables

rcTrucks : set of GarbageTruck := {};

sortedGarbage : GarbageSorter `GarbageMap := {
    <GLASSID>    |-> {},
    <METALID>    |-> {},
    <PAPERID>    |-> {},
    <PLASTICID>  |-> {}
}

```

```

};

operations

public Step : () ==> ()
Step() == (
  if card rcTrucks > 0 then
  (
    dcl sortedMap : GarbageSorter`GarbageMap := sortAllTrucks();
    handleGarbageMap(sortedMap);
  )
);

public addFilledTrucksToPlant : set of GarbageTruck ==> ()
addFilledTrucksToPlant(filledTruck) ==
  rcTrucks := rcTrucks union filledTruck;

private handleGarbageMap : GarbageSorter`GarbageMap ==> ()
handleGarbageMap(gpMap) ==
(
  IO`printf("Sorted Garbage at time %s: \r\n", [World`timer.GetTime()]);
  for all gptype in set dom gpMap do
  (
    cases gptype:
      <GLASSID> -> IO`println("Glass: "),
      <METALID> -> IO`println("Metal: "),
      <PAPERID> -> IO`println("Paper: "),
      <PLASTICID> -> IO`println("Plastic: "),
      others -> skip
    end;

    -- Set can abstract the maps set as a Large GarbagePack, therefore these functions work
    IO`print("Weight ");
    IO`println(GLOBAL`SumWeightGarbagePack(gpMap(gptype)));
    IO`print("Volume ");
    IO`println(GLOBAL`SumVolumeGarbagePack(gpMap(gptype)));
    IO`println("");
    sortedGarbage(gptype) := {};
  );
);

private sortAllTrucks : () ==> GarbageSorter`GarbageMap
sortAllTrucks() == (
  for all t in set rcTrucks
  do
  (
    let x = getSetOfIndividualGarbageFromTruck(t) in
      sortedGarbage := GarbageSorter`MapCombine(sortedGarbage, GarbageSorter`
        sortSetOfGarbageType(x));
    t.emptyTruck();
    GarbageSortingSystem`garbageSortingController.addTrucks({t});
    rcTrucks := rcTrucks \ {t};
  );
  return sortedGarbage
);

functions

private getSetOfIndividualGarbageFromTruck : GarbageTruck -> set of GarbageType

```

```

getSetOfIndividualGarbageFromTruck(gbTruck) == (
    dunion {i.getGarbagePack() | i in set gbTruck.getTruckGarbage()}
);

end RecyclingPlant

```

| Function or operation              | Line | Coverage | Calls |
|------------------------------------|------|----------|-------|
| Step                               | 16   | 100.0%   | 30    |
| addFilledTrucksToPlant             | 25   | 100.0%   | 20    |
| getSetOfIndividualGarbageFromTruck | 69   | 100.0%   | 75    |
| handleGarbageMap                   | 29   | 97.6%    | 48    |
| sortAllTrucks                      | 53   | 100.0%   | 50    |
| RecyclingPlant.vdmpp               |      | 98.9%    | 223   |

## 13 GarbageSortingTest

```

class GarbageSortingTest is subclass of GLOBAL, TestCase

values

gP1 : GarbagePack = new GarbagePack({new Metal(mk_dimensionsType(5,8,7), 100), -- W: 543, V:
    280+36+30+224+56+32 = 658
    new Paper(mk_dimensionsType(3,4,3), 140),
    new Glass(mk_dimensionsType(6,5,1), 38),
    new Plastic(mk_dimensionsType(7,8,4), 65),
    new Metal(mk_dimensionsType(7,2,4), 35),
    new Paper(mk_dimensionsType(1,8,4), 165)
});

gP2 : GarbagePack = new GarbagePack({new Metal(mk_dimensionsType(5,8,7), 170), -- W: 448, V:
    280+36+30 = 346
    new Paper(mk_dimensionsType(3,4,3), 140),
    new Glass(mk_dimensionsType(6,5,1), 138)
});

gP3 : GarbagePack = new GarbagePack({new Metal(mk_dimensionsType(1,2,5), 110), -- W: 414, V:
    10+14+2+64+7+12 = 109
    new Paper(mk_dimensionsType(7,2,1), 40),
    new Glass(mk_dimensionsType(1,2,1), 13),
    new Plastic(mk_dimensionsType(1,8,8), 85),
    new Metal(mk_dimensionsType(7,1,1), 31),
    new Paper(mk_dimensionsType(1,3,4), 135)
});

gP4 : GarbagePack = new GarbagePack({new Metal(mk_dimensionsType(5,8,7), 190), -- W: 527, V:
    280+36+30 = 346
    new Paper(mk_dimensionsType(3,4,3), 149),
    new Glass(mk_dimensionsType(6,5,1), 188)
});

sortedGarbage : GarbageSorter`GarbageMap = {    <GLASSID>    |-> {},
                                                <METALID>    |-> {},
                                                <PAPERID>   |-> {},
                                                <PLASTICID> |-> {}

```

```

};

-- Weights:
-- Metal = 100 + 35 + 170 + 110 + 31 + 190 = 636
-- Paper = 140 + 165 + 140 + 40 + 135 + 149 = 769
-- Glass = 38 + 138 + 13 + 188 = 377
-- Plastic = 65 + 85 = 150
-- == 1932

-- Volume:
-- Metal = 280 + 56 + 280 + 10 + 7 + 280 = 913
-- Paper = 36 + 32 + 36 + 14 + 12 + 36 = 166
-- Glass = 30 + 30 + 2 + 30 = 92
-- Plastic = 224 + 64 = 288
-- == 1459

operations

public GarbageSortingTest: seq of char ==> GarbageSortingTest
GarbageSortingTest(name_) ==
(name := name_);

protected SetUp : () ==> ()
SetUp() == skip;

protected TearDown: () ==> ()
TearDown() == skip;

protected RunTest: () ==> ()
RunTest() ==
(
  decl gs : GarbageSorter := new GarbageSorter(),
    sortedMap : GarbageSorter`GarbageMap,
    emptySet : set of GarbageType := {},
    metalWeight : nat := 0,
    paperWeight : nat := 0,
    glassWeight : nat := 0,
    plasticWeight : nat := 0,
    metalVol : nat := 0,
    paperVol : nat := 0,
    glassVol : nat := 0,
    plasticVol : nat := 0;

  emptySet := dunion {i.getGarbagePack() | i in set {gP1, gP2, gP3, gP4}};

  sortedMap := GarbageSorter`MapCombine(sortedGarbage, GarbageSorter`sortSetofGarbageType(
    emptySet));

  metalWeight := SumWeightGarbagePack(sortedMap(<METALID>));
  paperWeight := SumWeightGarbagePack(sortedMap(<PAPERID>));
  glassWeight := SumWeightGarbagePack(sortedMap(<GLASSID>));
  plasticWeight := SumWeightGarbagePack(sortedMap(<PLASTICID>));

  metalVol := SumVolumeGarbagePack(sortedMap(<METALID>));
  paperVol := SumVolumeGarbagePack(sortedMap(<PAPERID>));
  glassVol := SumVolumeGarbagePack(sortedMap(<GLASSID>));
  plasticVol := SumVolumeGarbagePack(sortedMap(<PLASTICID>));

  AssertTrue(metalWeight = 636);
  AssertTrue(paperWeight = 769);
  AssertTrue(glassWeight = 377);
  AssertTrue(plasticWeight = 150);

```

```

    AssertTrue(metalVol = 913);
    AssertTrue(paperVol = 166);
    AssertTrue(glassVol = 92);
    AssertTrue(plasticVol = 288);

    AssertTrue(metalWeight + paperWeight + glassWeight + plasticWeight = 1932);
    AssertTrue(metalVol + paperVol + glassVol + plasticVol = 1459);
}

end GarbageSortingTest

```

| Function or operation    | Line | Coverage | Calls |
|--------------------------|------|----------|-------|
| GarbageSortingTest       | 52   | 0.0%     | 0     |
| RunTest                  | 62   | 89.1%    | 1     |
| SetUp                    | 56   | 100.0%   | 1     |
| TearDown                 | 59   | 100.0%   | 1     |
| GarbageSortingTest.vdmpp |      | 94.3%    | 3     |

## 14 GarbageTruckTest

```

class GarbageTruckTest is subclass of GLOBAL, TestCase

values
gP1 : GarbagePack = new GarbagePack({new Metal(mk_dimensionsType(5,8,7), 100), -- 643
                                     new Paper(mk_dimensionsType(3,4,3), 140),
                                     new Glass(mk_dimensionsType(6,5,1), 138),
                                     new Plastic(mk_dimensionsType(7,8,4), 65),
                                     new Metal(mk_dimensionsType(7,2,4), 35),
                                     new Paper(mk_dimensionsType(1,8,4), 165)
                                   });

gP2 : GarbagePack = new GarbagePack({new Metal(mk_dimensionsType(5,8,7), 170), -- 448
                                     new Paper(mk_dimensionsType(3,4,3), 140),
                                     new Glass(mk_dimensionsType(6,5,1), 138)
                                   });

operations

public GarbageTruckTest: seq of char ==> GarbageTruckTest
GarbageTruckTest(name_) ==
(name := name_);

protected SetUp : () ==> ()
SetUp() == skip;

protected TearDown: () ==> ()
TearDown() == skip;

protected RunTest: () ==> ()
RunTest() ==
(

```

```

    dcl truck1 : GarbageTruck := new GarbageTruck("ID12");
    AssertTrue(truck1.isTruckFull() = false);
    truck1.addToTruck(gP1);
    AssertTrue(truck1.getTruckGarbage() = {gP1});
    AssertTrue(truck1.isTruckFull() = false);
    AssertTrue(GarbageTruck `willBeOverfilled(truck1.getTruckGarbage(), gP2.getPackWeight(),
        gP2.getPackVolume()) = true);
);
end GarbageTruckTest

```

| Function or operation  | Line | Coverage | Calls |
|------------------------|------|----------|-------|
| GarbageTruckTest       | 19   | 100.0%   | 1     |
| RunTest                | 29   | 100.0%   | 1     |
| SetUp                  | 23   | 100.0%   | 1     |
| TearDown               | 26   | 100.0%   | 1     |
| GarbageTruckTest.vdmpp |      | 100.0%   | 4     |

## 15 TRunner

```

class TRunner

operations

public Run: () ==> ()
Run () ==
(
    let t : TestSuite = new TestSuite(), result = new TestResult()
    in
    (
        t.AddTest(new GarbageTruckTest("Truck unittest"));
        t.AddTest(new GarbageSortingTest());
        t.Run(result);
        result.Show();
    );
)
end TRunner

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| Run                   | 4    | 100.0%   | 1     |
| TRunner.vdmpp         |      | 100.0%   | 1     |

## 16 Test

```

class Test

operations

public Run: TestResult ==> ()

```



```

    Run (-) == is subclass responsibility
end Test

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| Run                   | 4    | 100.0%   | 8     |
| Test.vdmpp            |      | 100.0%   | 8     |

## 17 TestCase

```

class TestCase
  is subclass of Test

instance variables
  protected name : seq of char

operations

  public TestCase: seq of char ==> TestCase
    TestCase(nm) == name := nm;

  public GetName: () ==> seq of char
    GetName () == return name;

  protected AssertTrue: bool ==> ()
    AssertTrue (pb) == if not pb then exit <FAILURE>;

  protected AssertFalse: bool ==> ()
    AssertFalse (pb) == if pb then exit <FAILURE>;

  public Run: TestResult ==> ()
    Run (ptr) ==
      trap <FAILURE>
      with
        ptr.AddFailure(self)
      in
        (SetUp());
    RunTest();
    TearDown();

  protected SetUp: () ==> ()
    SetUp () == is subclass responsibility;

  protected RunTest: () ==> ()
    RunTest () == is subclass responsibility;

  protected TearDown: () ==> ()
    TearDown () == is subclass responsibility
end TestCase

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| AssertFalse           | 17   | 0.0%     | 0     |
| AssertTrue            | 14   | 60.0%    | 0     |
| GetName               | 11   | 0.0%     | 0     |
| Run                   | 20   | 71.4%    | 2     |
| RunTest               | 33   | 100.0%   | 8     |
| SetUp                 | 30   | 100.0%   | 8     |
| TearDown              | 36   | 100.0%   | 8     |
| TestCase              | 8    | 0.0%     | 0     |
| TestCase.vdmpp        |      | 47.8%    | 26    |

## 18 TestResult

```

class TestResult

instance variables
  failures : seq of TestCase := []

operations

  public AddFailure: TestCase ==> ()
  AddFailure (ptst) == failures := failures ^ [ptst];

  public Print: seq of char ==> ()
  Print (pstr) ==
    def - = new IO().echo(pstr ^ "\n") in skip;

  public Show: () ==> ()
  Show () ==
    if failures = [] then
      Print ("No failures detected")
    else
      for failure in failures do
        Print (failure.GetName() ^ " failed")

end TestResult

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| AddFailure            | 7    | 0.0%     | 0     |
| Print                 | 10   | 100.0%   | 1     |
| Show                  | 14   | 46.1%    | 1     |
| TestResult.vdmpp      |      | 55.5%    | 2     |

## 19 TestSuite

```
class TestSuite
  is subclass of Test

instance variables
  tests : seq of Test := [];

operations

public Run: () ==> ()
Run () ==
  (dcl ntr : TestResult := new TestResult ();
   Run(ntr);
   ntr.Show());

public Run: TestResult ==> ()
Run (result) ==
  for test in tests do
    test.Run(result);

public AddTest: Test ==> ()
AddTest(test) ==
  tests := tests ^ [test];

end TestSuite
```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| AddTest               | 19   | 100.0%   | 2     |
| Run                   | 8    | 100.0%   | 2     |
| TestSuite.vdmpp       |      | 62.5%    | 4     |

## 20 Environment

```
class Environment is subclass of GLOBAL

types

inline = seq of char * seq of gbpackinputtype * Time

instance variables
FinishedCollecting : bool := false;

inlines      : seq of inline := [];

operations

public Run : () ==> ()
Run () ==
  (
    while (not FinishedCollecting) do
      (
        updateAddresses();
```

```

        GarbageSortingSystem`garbageSortingController.Step();
        GarbageSortingSystem`plant.Step();
        World`timer.StepTime();
    );
);

private updateAddresses : () ==> ()
updateAddresses() ==
(
    if len inlines > 0
    then
        (dcl curtime : Time := World`timer.GetTime(),
         doneRead : bool := false;
         while not doneRead do
             (
                 def mk_(adrString, gbpckinput, objtime) = hd inlines
                 in
                     if objtime <= curtime
                     then (
                         dcl gtset : set of GarbageType := {};
                         for gps in gbpckinput
                         do
                             (
                                 cases gps.#1:
                                 <GLASSID> -> gtset:= gtset union {new Glass(mk_dimensionsType(gps
                                     .#2.width,gps.#2.length,gps.#2.height), gps.#3)},
                                 <METALID> -> gtset:= gtset union {new Metal(mk_dimensionsType(gps
                                     .#2.width,gps.#2.length,gps.#2.height), gps.#3)},
                                 <PAPERID> -> gtset:= gtset union {new Paper(mk_dimensionsType(gps
                                     .#2.width,gps.#2.length,gps.#2.height), gps.#3)},
                                 <PLASTICID> -> gtset:= gtset union {new Plastic(mk_dimensionsType
                                     (gps.#2.width,gps.#2.length,gps.#2.height), gps.#3)},
                                 others -> skip
                             end;
                         );
                         GarbageSortingSystem`addressRepository.addToAddresses({adrString |-> {new
                             GarbagePack(gtset)}});
                         inlines := tl inlines;
                         doneRead := len inlines = 0;
                     )
                     else
                         doneRead := true
                 )
             )
        )
    else
        FinishedCollecting := true
);

public Environment : seq of char ==> Environment
Environment(fname) ==
(
    def mk_(-,input) = IO`freadval[seq of inline](fname)
    in
        inlines := input;
)
end Environment

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
|-----------------------|------|----------|-------|

|                   |    |        |    |
|-------------------|----|--------|----|
| Environment       | 64 | 100.0% | 7  |
| Run               | 14 | 100.0% | 20 |
| updateAddresses   | 26 | 99.2%  | 15 |
| Environment.vdmpp |    | 99.2%  | 42 |

## 21 GLOBAL

```

class GLOBAL

instance variables

public static GARBAGETYPE_MAX_WEIGHT      : nat := 200;
public static GARBAGETYPE_MAX_WIDTH       : nat := 20;
public static GARBAGETYPE_MAX_LENGTH      : nat := 15;
public static GARBAGETYPE_MAX_HEIGHT      : nat := 30;
public static GARBAGETRUCK_MAX_WEIGHT     : nat := 1000;
public static GARBAGEPACK_MAX_NR          : nat := 10;
public static GARBAGETRUCK_MAX_VOLUME     : nat := GARBAGETYPE_MAX_WIDTH * GARBAGETYPE_MAX_LENGTH *
    GARBAGETYPE_MAX_HEIGHT * GARBAGEPACK_MAX_NR;

public static InvalidAddressChars : set of char := {'!', '#', '%', '&', '/', '(', ')', '=',
    '^', '*', '.', '<', '>'};
public static allowedIdNbrs : set of char := {'1', '2', '3', '4', '5', '6', '7', '8', '9'};

types
    public Time = nat;

    public gbpackinputtype = GarbageId * dimensionsType * nat;

    public GarbageId = <GLASSID> | <METALID> | <PAPERID> | <PLASTICID>;

    public dimensionsType :: width : nat
                             length : nat
                             height : nat;

    public GarbageMap = map GarbageId to set of GarbageType;

functions

public SumVolumeGarbagePack: set of GarbageType +> nat
SumVolumeGarbagePack(s) ==
    if s = {}
    then 0
    else let e in set s in
        e.getVolume() + SumVolumeGarbagePack(s \ {e})
measure card s;

public SumWeightGarbagePack: set of GarbageType +> nat
SumWeightGarbagePack(s) ==
    if s = {}
    then 0
    else let e in set s in
        e.getWeight() + SumWeightGarbagePack(s \ {e})
measure card s;

```

```

public SumSet: set of nat -> nat
SumSet(s) ==
  if s = {}
  then 0
  else let e in set s in
    e + SumSet(s \ {e})
measure card s;

end GLOBAL

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| SumSet                | 51   | 100.0%   | 444   |
| SumVolumeGarbagePack  | 34   | 100.0%   | 570   |
| SumWeightGarbagePack  | 42   | 100.0%   | 1246  |
| GLOBAL.vdmpp          |      | 100.0%   | 2260  |

## 22 Timer

```

class Timer

instance variables
  currentTime : nat := 0;

values
  stepLength : nat = 100;

operations

public

  StepTime: () ==> ()
  StepTime() ==
    currentTime := currentTime + stepLength;

public

  GetTime: () ==> nat
  GetTime() == return currentTime;

end Timer

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| GetTime               | 17   | 100.0%   | 30    |
| StepTime              | 12   | 100.0%   | 20    |
| Timer.vdmpp           |      | 100.0%   | 50    |

## 23 World

```
class World
instance variables

public static
  env : [Environment] := nil;

public static
  timer : Timer := new Timer();

operations

public

  World : () ==> World
  World() ==
  (
    env := new Environment("scenario.txt");
    GarbageSortingSystem`garbageSortingController.addTrucks(GarbageSortingSystem`trucks)
  );

public

  Run : () ==> ()
  Run() ==
    env.Run();

end World
```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| Run                   | 21   | 100.0%   | 5     |
| World                 | 13   | 100.0%   | 7     |
| World.vdmpp           |      | 100.0%   | 12    |