1.124J Foundations of Software Engineering

Problem Set 3 - Solution

Due Date: Tuesday 10/3/00

Problem 1:[100%]

makeSol3:

```
#!gmake
#============================================================
#
#          Makefile for Problem Set # 3
#!gmake
#
# To use this makefile:    % gmake -f makeSol3 program_name
#
#      Fall - 2000
#
#=============================================================

#     V a r i a b l e    D e f i n i t i o n s
#     ----------------------------------------
MACHINE_TYPE = `/bin/athena/machtype`
CXX = g++

CXXINCLUDE = -I.
CXXFLAGS = -g -ansi -pedantic -Wall
LDLIBS =  -lm

SRC  = sol3.C property.C vehicle.C motorvehicle.C car.C motorcycle.C bike.C vehicleNode.C
PROG = sol3
OBJ =  $(SRC:%.C=%.o)

#          E x p l i c i t     R u l e s
#          ------------------------------
#------------------------------------------------------------
all: ${PROG}
.PHONY: all
${PROG}: makeSol3
${OBJ}: makeSol3
#-----------------------------------------------------------------
sol3: sol3.o  property.o vehicle.o motorvehicle.o \
     car.o motorcycle.o bike.o vehicleNode.o
 @ echo "     Linking  to create $@"
 $(CXX) sol3.o   property.o vehicle.o motorvehicle.o  car.o \
            motorcycle.o bike.o vehicleNode.o -o sol3  ${LDLIBS}



sol3.o:sol3.C sol3.h property.h vehicle.h motorvehicle.h \
     car.h motorcycle.h bike.h vehicleNode.h
 @ echo "     Compiling $< to create $@ "
 $(CXX) ${CXXFLAGS}  -c sol3.C



vehicle.o:vehicle.C vehicle.h
 @ echo "     Compiling $< to create $@ "
 $(CXX) ${CXXFLAGS}  -c vehicle.C

motorvehicle.o:motorvehicle.C motorvehicle.h  vehicle.h
 @ echo "     Compiling $< to create $@ "
 $(CXX) ${CXXFLAGS}  -c motorvehicle.C

car.o:car.C car.h vehicle.h motorvehicle.h property.h
 @ echo "     Compiling $< to create $@ "
 $(CXX) ${CXXFLAGS}  -c car.C

motorcycle.o:motorcycle.C motorcycle.h vehicle.h motorvehicle.h property.h
 @ echo "     Compiling $< to create $@ "
 $(CXX) ${CXXFLAGS}  -c motorcycle.C

bike.o:bike.C bike.h vehicle.h  property.h
 @ echo "     Compiling $< to create $@ "
 $(CXX) ${CXXFLAGS}  -c bike.C



vehicleNode.o:vehicleNode.C vehicleNode.h vehicle.h motorvehicle.h
```

```
 @ echo "     Compiling $< to create $@ "
 $(CXX) ${CXXFLAGS}  -c vehicleNode.C

#-----------------------------------------------------------------
.PHONY: clean clean_o clean_p
clean:
 @echo "     Cleaning all executable and object files"
 -rm -f $(PROG) *.o a.out
clean_o:
 @echo "     Cleaning all object files"
 -rm  -f *.o
clean_p:
 @echo "     Cleaning all executables"
 -rm -f $(PROG)

#---------------------------------------------------------------


#          I m p l i c i t      R u l e s
#          -------------------------------
#-----------------------------------------------------------------
%: %.o
 @ echo "     Linking  to create $@"
 $(CXX) $<  -o   $@ ${LDLIBS}
%.o:%.C
 @ echo "     Compiling $< to create $@ "
 $(CXX) ${CXXFLAGS}  -c $< -o $@
#---------------------------------------------------------------
```

sol3.h:

```
// Problem Set#3 - solution  [sol3.h]

#ifndef SOL3_H
#define SOL3_H

////////////////////////////////////////

#include "vehicle.h"
```

```
#include "motorvehicle.h"

#include "property.h"

#include "car.h"

#include "motorcycle.h"

#include "bike.h"

#include "car.h"

#include "vehicleNode.h"

////////////////////////////////////////

int main(void);

void printMenu();

char getSelection();

void processSelection(char selection);
// To process the selected option

void continueStep(void);
// To delay the clearing of the screen

void showVehicles();
// shows the current vehicles

void printNumberOfVehicles();
// prints the number of vehicles

void saveToFiles(void);
// Saves the data to a file



void printMenu(char *str);
// prints the menu for
// data addition/removal


void addVehicle(char selection);
// Add vehicle data

void processAddition(char selection);
// Process data addition

void addCar();
//  Adds a car

void addMotorcycle();
// Adds a motorcycle

void addBike();
// Adds a bike

void getPropertyData(double &value, char *admin);
// Gets Property-related data from the user

void getMotorvehicleData(int & id, double &weight, char *brand,
                  char *model, char *type, int &cc, int &hp);
// Gets Motorvehicle-related data from the user

void getVehicleData(int & id, double &weight, char *brand,
                              char *model, char *type);
// Gets Vehicle-related data from the user


void releaseMemory(void);
// Frees all dynamically allocated memory

#endif
```

sol3.C:

```
  // Problem Set#3 - Solution  [sol3.C]
```

```
#include <stdlib.h>
#include <iomanip.h>
#include <string.h>
#include <iostream.h>
#include <fstream.h>

#include "sol3.h"

/********** Static initializations ***********/
VehicleNode* VehicleNode::head = NULL;

int Vehicle::numberVehicles = 0;
int Motorvehicle::numberMotorvehicles = 0;

int Car::numberCars = 0;
ofstream Car::outputFile;

int Motorcycle::numberMotorcycles = 0;
ofstream Motorcycle::outputFile;

int Bike::numberBikes = 0;
ofstream Bike::outputFile;

/*********************************************/

/***************************************************************/
int main()
{
  char selection;

  do
    {
      printMenu();
      selection = getSelection();
      processSelection(selection);
    }while(selection != 'q' && selection != 'Q');

  releaseMemory();

  return EXIT_SUCCESS;
}
```

```
/*****************************************************************/




/*********************************************************************/
void printMenu()
{
  system("clear");
  cout << "**************************************************" << endl;
  cout << "\n [A]: Add a vehicle\n" << endl;
  cout << "\n [S]: Show vehicles \n" << endl;
  cout << "\n [N]: Number of vehicles \n" << endl;
  cout << "\n [F]: Save vehicles to files\n" << endl;
  cout << "\n [Q]: Quit\n" << endl;
  cout << "**************************************************" << endl;
  cout << "\t Your selection: " ;
}
/*********************************************************************/

/*********************************************************************/
void printMenu(char *str)
{
  system("clear");
  cout << "**************************************************" << endl;
  cout << "\n [C]: " << str << " a car\n" << endl;
  cout << "\n [M]: " << str << " a motorcycle\n" << endl;
  cout << "\n [B]: " << str << " a bike\n" << endl;
  cout << "\n [Q]: Quit operation \n" << endl;
  cout << "**************************************************" << endl;
  cout << "\t Your selection: " ;
}
/*********************************************************************/



/*****************/
char getSelection()
{
  char selection;

  cin >> selection;

  return selection;
```

```
}
/*****************/



/***********************************************************/
void processSelection(char selection)
{

  switch(selection)
    {
    case 'A':  case 'a':
      addVehicle(selection);
      break;

    case 'S':  case 's':
      showVehicles();
      break;

    case 'N':  case 'n':
      printNumberOfVehicles();
      break;

    case 'F':  case 'f':
      saveToFiles();
      break;

    case 'Q':  case 'q':
      break;

    default:
      cout << "Imporper Selection: Please select again" << endl;
    }

  continueStep();
}
/***************************************************************/




/*******************************************************/
void continueStep(void)
{
  char str[20];

  cout << "\n Press any button and the <Enter> to proceed... ";
  cin >> str;
}
/********************************************************/



/******************************************************/
void addVehicle(char selection)
{
  printMenu("Add");
  selection = getSelection();
  processAddition(selection);
}
/*****************************************************/




/****************************************************************/
void processAddition(char selection)
{
  switch(selection)
    {
    case 'C':  case 'c':
      addCar();
      break;

    case 'M':  case 'm':
      addMotorcycle();
      break;

    case 'B':  case 'b':
      addBike();
      break;

    case 'Q':  case 'q':
      break;

    default:
      cout << "Imporper Selection: Please select again" << endl;
    }
}
/*****************************************************************/



/***********************************************************/
void addCar()
{
  char brand[40], model[40], type[40], admin[40];
  int id, cc, hp, seats, airbags;
  double weight, value;

  cout << " Adding a car....\n "<< endl;
  cout << "   Please give all the relevant information" << endl;

  getMotorvehicleData(id, weight, brand, model, type, cc, hp);

  getPropertyData(value, admin);

  cout << "\n Number of seats = " ;
  cin >> seats;
  cout << " Number of airbags = " ;
  cin >> airbags;


  Vehicle *v = new Car( id, weight, brand,  model, type, cc,
                        hp, seats, airbags, value, admin);


  VehicleNode::getHead() -> add(new VehicleNode(v));
}
/**************************************************************/

/*****************************************************************/
void getVehicleData(int & id, double &weight, char *brand,
                               char *model, char *type)
{
  cout << "\n Vehicle ID = " ;
  cin >> id;
  cout << " Weight = " ;
  cin >> weight;
  cout << " Brand = " ;
  cin >> brand;
  cout << " Model = " ;
  cin >> model;
  cout << " Type = " ;
  cin >> type;
}
/*****************************************************************/

/*****************************************************************/
void getMotorvehicleData(int & id, double &weight, char *brand,
                    char *model, char *type, int &cc, int &hp)
{
  getVehicleData(id, weight, brand, model, type);
  cout << "\n CC = " ;
  cin >> cc;
  cout << " hp = " ;
  cin >> hp;

}
/*****************************************************************/

/*****************************************************************/
void getPropertyData(double &value, char *admin)
{
  cout << "\n Estimated value = " ;
  cin >> value;
  cout << " Administrator = " ;
  cin >> admin;
}
/*****************************************************************/

/*****************************************************************/
void addMotorcycle()
{
  char brand[40], model[40], type[40], admin[40];
  int id, cc, hp, strokes;
  double weight, value;

  cout << " Adding a motorcycle....\n "<< endl;
  cout << "   Please give all the relevant information" << endl;

  getMotorvehicleData(id, weight, brand, model, type, cc, hp);
  getPropertyData(value, admin);

  cout << "\n Number of strokes = " ;
  cin >> strokes;

  Vehicle *v = new Motorcycle(id, weight, brand,  model, type,
                              cc, hp, strokes, value, admin);

  VehicleNode::getHead() -> add(new VehicleNode(v));
}
/*****************************************************************/



/*****************************************************************/
void addBike()
{
  char brand[40], model[40], type[40], admin[40], c;
  int id;
  double weight, value;
  bool horn=false, lights=false;

  cout << " Adding a bike....\n "<< endl;
  cout << "   Please give all the relevant information" << endl;

  getVehicleData(id, weight, brand, model, type);

  getPropertyData(value, admin);

  do
    {
      cout << "\n Horn [Y/N] = " ;
      cin >> c;
    }while(c!='Y' && c!='y' && c!='N' && c!='n');
  if( c!='N' && c!='n')
    horn=true;

  do
    {
      cout << "\n Lights [Y/N] = " ;
      cin >> c;
    }while(c!='Y' && c!='y' && c!='N' && c!='n');
  if( c!='N' && c!='n')
    lights=true;



  Vehicle *v = new Bike(id, weight, brand,  model, type,
                                horn, lights, value, admin);

  VehicleNode::getHead() -> add(new VehicleNode(v));
}
/****************************************************************/



/***********************************/
void showVehicles()
{
  cout << "\n Showing vehicles' data...\n " << endl;
  VehicleNode::show() ;
}
/***********************************/



/
***********************************************************************************/
void printNumberOfVehicles()
{
  cout << "Number of vehicles: " << Vehicle::getNumberVehicles() << endl;
  cout << "  Number of motorvehicles: " << Motorvehicle::getNumberMotorvehicles() << endl;
  cout << "     Number of cars: " << Car::getNumberCars() << endl;
  cout << "     Number of motorcycles: " << Motorcycle::getNumberMotorcycles() << endl;
  cout << "  Number of bikes: " << Bike::getNumberBikes() << endl << endl;
}
/
***********************************************************************************/



/**************************************/
void saveToFiles(void)
{
  VehicleNode::save() ;
}
/**************************************/
```

```
/*************************************/
void releaseMemory()
{
  cout << "releasing memory " << endl;

  VehicleNode::freeMemory();
}
/**************************************/
```

Vehicle.h:

```
// Problem Set#3 - Solution [vehicle.h]

#ifndef Vehicle_H
#define Vehicle_H

class Vehicle
{
private:
  int idNumber;
  double weight;
  char *brand;
  char *model;
  char *type;
  static int numberVehicles;



public:
  Vehicle(int id, double w, char *brand,  char *model, char *type);
  virtual ~Vehicle();

  virtual void show(void);
  virtual void save()=0;
  virtual void save(ofstream &o);

  static int getNumberVehicles(void)
    {
      return numberVehicles;
    }
};


#endif
```

Vehicle.C:

```
//  Problem Set#3 -  Solution [vehicle.C]

#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <fstream.h>
#include <string.h>
#include <stdio.h>
#include <fstream.h>
#include "vehicle.h"



Vehicle::Vehicle(int id, double w, char *brand,  char *model,  char *type)
{
  idNumber = id;
  weight = w;
  this->brand = new char[strlen(brand)+1];
  strcpy(this->brand,brand);
  this->model = new char[strlen(model)+1];
  strcpy(this->model,model);
  this->type = new char[strlen(type)+1];
  strcpy(this->type,type);
  numberVehicles++;
}

Vehicle::~Vehicle()
{
  cout << "Deleting a Vehicle obejct" << endl;
  numberVehicles--;
  delete [] brand;
  delete [] model;
  delete [] type;
}
```

```
void Vehicle::show()
{
  cout  << " ID number= "  << idNumber << "   Brand = " << brand
 << "   Model = " << model << "   Type = " << type
      << "   Weight = "  << weight  << endl;
}

void Vehicle::save(ofstream &o)
{
  o << setw(10) << brand << setw(10) << model << setw(10) << type
    << setw(5) << idNumber << setw(10) << weight ;
}
```

---

motorvehicle.h:

```
// Problem Set#3 - Solution [motorVehicle.h]

#ifndef MotorVehicle_H
#define MotorVehicle_H

#include "vehicle.h"

class Motorvehicle : public Vehicle
{
private:
  int cc;
  int hp;
  static int numberMotorvehicles;

public:
  Motorvehicle(int id, double weight, char *brand,  char *model,
                              char *type, int cc, int hp);
  virtual ~Motorvehicle();
  virtual void show(void);
  virtual void save(ofstream &o);
  static int getNumberMotorvehicles(void)
    {
      return numberMotorvehicles;
    }
};
```

```
#endif
```

motorvehicle.C:

```
//  Problem Set#3 -  Solution [motorvehicle.C]

#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <fstream.h>
#include <string.h>
#include <stdio.h>
#include <fstream.h>

#include "motorvehicle.h"


Motorvehicle::Motorvehicle
(int id, double weight, char *brand,  char *model, char *type, int cc, int hp)
:Vehicle(id,weight,brand,model,type)
{
  this -> cc = cc ;
  this -> hp = hp ;
  numberMotorvehicles++;
}

Motorvehicle::~Motorvehicle()
{
  cout << "Deleting a Motorvehicle obejct" << endl;
  numberMotorvehicles--;
}

void Motorvehicle::show()
{
  Vehicle::show();
  cout << " CC = " << cc << "   Horsepower = " << hp << endl;
}

void Motorvehicle::save(ofstream &o)
{
  Vehicle::save(o);
  o << setw(6) << cc << setw(5) << hp ;
}
```

---

car.h:

```
// Problem Set#3 - Solution [car.h]

#ifndef Car_H
#define Car_H

#include "vehicle.h"
#include "motorvehicle.h"
#include "property.h"

class Car : public Motorvehicle, private Property
{
private:
  int seats;
  int airBags;
  static int numberCars;
  static ofstream outputFile;

public:
  Car(int id, double weight, char *brand,  char *model, char *type,
        int cc, int hp, int seats, int airBags, double value, char *admin);
  virtual ~Car();
  virtual void show(void);
  virtual void save(void);
  static void commentOutputFile();
  static int getNumberCars(void)
    {
      return numberCars;
    }
};
```

```
#endif
```

car.C:

```
//  Problem Set#3 -  Solution [car.C]

#include <iomanip.h>
#include <stdlib.h>
#include <fstream.h>
#include <string.h>
#include <stdio.h>
#include <fstream.h>
#include "car.h"



Car::Car(int id, double weight, char *brand,  char *model, char *type,
        int cc, int hp, int seats, int airBags, double value, char *admin)
:Motorvehicle(id,weight,brand,model,type,cc,hp), Property(value,admin)
{
  Car::seats = seats;
  Car::airBags = airBags;
  numberCars++;
}



Car::~Car()
{
  cout << "Deleting a Car obejct" << endl;
  numberCars--;
}

void Car::show(void)
{
  cout << "  Car" << endl;
  Motorvehicle::show();
  cout<< " Seat =" << seats << "  Airbag = " << airBags << endl;
  Property::show();
}
```

```
void  Car::commentOutputFile()
{
  outputFile << " Cars " << endl;
}



void Car::save()
{
  static int i=0;
  if(i==0)
    {
      outputFile.open("cars.dat", ios::out);
      outputFile  << setw(10) << "Brand" << setw(10) << "Model"
    << setw(10) << "Type" << setw(8) << "ID"
    << setw(8) << "Weight"
    << setw(6) << "CC"   << setw(4) << "HP"
    << setw(6) << "Seats" << setw(9) << "Airbags"
    << setw(7) << "Value" << setw(16) << "Administrator"
    <<"\n--------------------------------------------"
    <<"---------------------------------------------"
    << endl;
    }

  outputFile << ++i ;
  Motorvehicle::save(outputFile);
  outputFile << setw(4) << seats << setw(5) << airBags;
  Property::save(outputFile);
  if(i==numberCars)
    {
      i=0;
      outputFile.close();
    }
}
```

---

motorcycle.h:

```
// Problem Set#3 - Solution [motorcycle.h]
```

```
#ifndef Motorcycle_H
#define Motorcycle_H

#include "vehicle.h"
#include "motorvehicle.h"
#include "property.h"



class Motorcycle : public Motorvehicle, private Property
{
private:
  int engineStrokes;
  static int numberMotorcycles;
  static ofstream outputFile;

public:
  Motorcycle(int id, double w, char *brand,  char *model, char *type,
                      int cc, int hp, int strokes, double value, char *admin);
  virtual ~Motorcycle();
  virtual void show(void);
  virtual void save(void);
  static void commentOutputFile();
  static int getNumberMotorcycles(void)
    {
      return numberMotorcycles;
    }
};



#endif
```

motorcycle.C:

```
   //  Problem Set#3 -  Solution [motorcycle.C]

#include <iomanip.h>
#include <stdlib.h>
#include <fstream.h>
```

```
#include <string.h>
#include <stdio.h>
#include <fstream.h>
#include "motorcycle.h"

Motorcycle::Motorcycle(int id, double weight, char *brand,  char *model, char *type,
                       int cc, int hp, int strokes, double value, char *admin)
:Motorvehicle(id,weight,brand,model,type,cc,hp), Property(value,admin)
{
  engineStrokes = strokes;
  numberMotorcycles++;
}

Motorcycle::~Motorcycle()
{
  cout << "Deleting a MotorCycle obejct" << endl;
  numberMotorcycles--;
}

void Motorcycle::show(void)
{
  cout << "  Motorcycle" << endl;
  Motorvehicle::show();
  cout << "Engine strokes = " << engineStrokes ;
  Property::show();
}



void Motorcycle::commentOutputFile()
{
  outputFile << " Motorcycle " << endl;
}



void Motorcycle::save(void)
{
  static int i=0;
  if(i==0)
    {
      outputFile.open("motorcycles.dat", ios::out);
      outputFile  << setw(10) << "Brand" << setw(10) << "Model"
    << setw(10) << "Type" << setw(8) << "ID"
    << setw(8) << "Weight"
    << setw(6) << "CC"   << setw(4) << "HP"
    << setw(9) << "Strokes"
    << setw(7) << "Value" << setw(15) << "Administrator"
    <<"\n------------------------------------------"
    <<"--------------------------------------------"
    << endl;
    }

  outputFile << ++i ;
  Motorvehicle::save(outputFile);
  outputFile << setw(5) << engineStrokes ;
  Property::save(outputFile);
  if(i==numberMotorcycles)
    {
      i=0;
      outputFile.close();
    }
}
```

---

bike.h:

```
  // Problem Set#3 - Solution [bike.h]

#ifndef Bike_H
#define Bike_H

#include "vehicle.h"
#include "property.h"

class Bike : public Vehicle, private Property
{
private:
  bool horn;
  bool lights;
  static int numberBikes;
  static ofstream outputFile;
public:
  Bike(int id, double weight, char *brand,  char *model, char *type,
                  bool horn, bool lights, double value, char *admin);
  virtual ~Bike();
  virtual void show();
  static void commentOutputFile();
  virtual void save(void);
  static int getNumberBikes(void)
    {
      return numberBikes;
    }
};



#endif
```

bike.C:

```
 //  Problem Set#3 -  Solution [bike.C]

#include <stdlib.h>
#include <iomanip.h>
#include <string.h>
#include <iostream.h>
#include <fstream.h>
#include "bike.h"

Bike::Bike(int id, double weight, char *brand,  char *model,
    char *type, bool horn, bool lights, double value, char *admin)
:Vehicle(id,weight,brand,model,type), Property(value,admin)
{
  this -> horn = horn ;
  this -> lights = lights ;
  numberBikes++;
}

Bike::~Bike()
```

```
{
  cout << "Deleting a Bike obejct" << endl;
  numberBikes--;
}

void Bike::show()
{
  cout << "  Bike" << endl;
  Vehicle::show();
  cout << " Has " <<((horn)?" Horn":" No Horn")
      << "  -    Has " << ((lights)?"Lights":" No Lights ")
       << endl ;
  Property::show();
}



void Bike::commentOutputFile()
{
  outputFile << " Bicycle " << endl;
}



void Bike::save(void)
{
  static int i=0;
  if(i==0)
    {
      outputFile.open("bikes.dat", ios::out);
      outputFile  << setw(10) << "Brand" << setw(10) << "Model"
    << setw(10) << "Type" << setw(8) << "ID"
    << setw(8) << "Weight"
    << setw(6) << "Horn" << setw(7) << "Lights"
    << setw(7) << "Value" << setw(15) << "Administrator"
    <<"\n-------------------------------------------"
    <<"--------------------------------------------"
    << endl;
    }

  outputFile << ++i ;
  Vehicle::save(outputFile);
  outputFile << setw(6) <<((horn)?"Yes":"No")
      << setw(7) << ((lights)?"Yes":"No") ;
  Property::save(outputFile);
  if(i==numberBikes)
    {
      i=0;
      outputFile.close();
    }
}
```

---

property,h:

```
// Problem Set#3 - Solution [property.h]

#ifndef Property_H
#define Property_H



class Property
{
private:
  double estimatedValue;
  char *administrator;



public:
  Property(double value, char admin[]);
  ~Property();

  void show();
  void save(ofstream &o);

};



#endif
```

property.C

```
//  Problem Set#3 -  Solution [property.C]

#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <fstream.h>
#include <string.h>
#include <stdio.h>
#include "property.h"


Property::Property(double value, char *admin)
{
  estimatedValue = value;
  administrator = new char[strlen(admin)+1];
  strcpy(administrator,admin);
}


Property::~Property()
{
  cout << "Deleting a Property obejct" << endl;
  delete [] administrator;
}

void Property::show()
{
  cout << " Estimated Value = " << estimatedValue
       << "   Administrator = " << administrator << endl;
}

void Property::save(ofstream &o)
{
  o << setw(12) << estimatedValue << setw(15) << administrator << endl;
}
```

---

vehicleNode.h:

```
// Problem Set#3 - Solution [vehicleNode.h]
```

```
#ifndef VehicleNode_H
#define VehicleNode_H

#include "vehicle.h"

class VehicleNode
{

 private:

  static VehicleNode *head;

  VehicleNode *previous;
  Vehicle *vehicle;
  VehicleNode *next;


 public:

   VehicleNode(Vehicle *p);

   ~VehicleNode();

   static VehicleNode* getHead(void);

   static void add(VehicleNode *p);

   static void show(void);

   static void save();

   static void freeMemory();

};



#endif
```

vehicleNode.C:

```
//  Problem Set#3 -  Solution [vehicleNode.C]

#include  <iostream.h>
#include  <stdlib.h>
#include <string.h>
#include <iomanip.h>
#include <fstream.h>
#include "vehicleNode.h"


/**********************************************************************/
                                        //   Constructor
VehicleNode::VehicleNode(Vehicle *v)
  {
    previous = NULL;
    vehicle = v;
    next = NULL;
  }
/**********************************************************************/



/***********************************************************************/
VehicleNode::~VehicleNode()         // Destructor
  {
    cout << "\nDestroying a VehicleNode object" <<  endl;
    delete vehicle;
  }
/***********************************************************************/




/**************  Member functions ************************/

/*************************************/
VehicleNode* VehicleNode::getHead(void)
{
  return head;
```

```
}
/************************************/

/***********************************/
void VehicleNode::add(VehicleNode *v)
{
  if(head==NULL)
    {
      head = v;
    }
  else
    {
      v->next = head;
      head->previous = v;
      head = v;
    }
}
/**********************************/



/***************************************************************/
void VehicleNode::show(void)
{
  VehicleNode *tmp = head;
  int i =0;

  if(!tmp)
    {
      cout << "\n\t\t No vehicles are in the list.\n" << endl;
    }
  else
    {
      cout << "\t Vehicles in the list"
    << "\n\t ---------------------\n" << endl;

      while(tmp)
 {
   cout << "\nVehicle " << ++i << ": ";
   tmp -> vehicle -> show();
   tmp = tmp->next;
   cout << "\n-----------------------------------"
        <<   "-----------------------------------" << endl;
 }
```

```
    }
}
/*************************************************************/




/**************************************************************/
void VehicleNode::save()
{
  VehicleNode *tmp = head;

  if(!tmp)
    {
      cout << "\n\t\t No vehicles are in the list.\n" << endl;
    }
  else
    {
      cout << "Vehicles are stored in individual files"
    << "\n " << endl;
      while(tmp)
 {
   tmp -> vehicle -> save();
   tmp = tmp->next;
 }
    }
}
/*************************************************************/



/**********************************************/
void VehicleNode::freeMemory()
{
  VehicleNode *tmp = head;

  if(head)
    {
      do
 {
   tmp = head -> next;
   delete head;
   head = tmp;
 }while(head);
```

```
      cout << "Memory has been released" << endl;
    }

}
/*****************************************************/
```