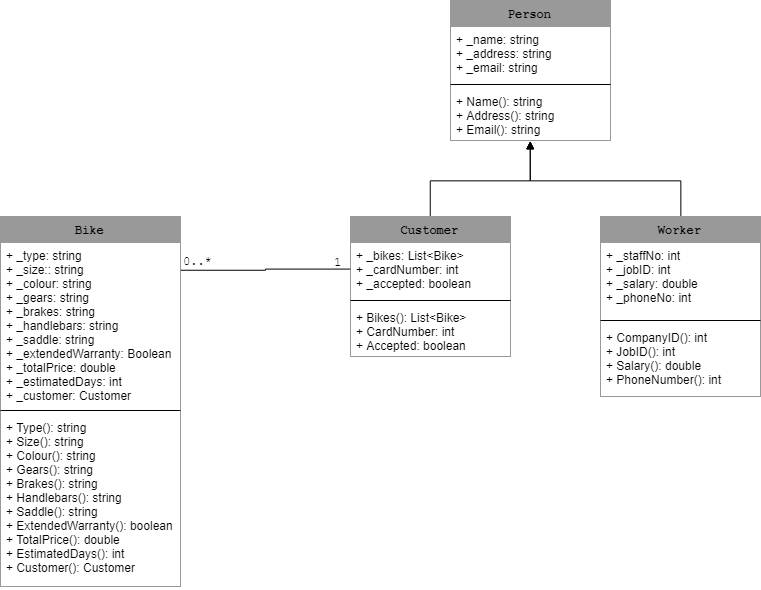
Software Engineering Methods Coursework

Part 2

# UML Class diagram



# Implementation and test units

## Classes:

|  |
| --- |
| namespace SEM\_cw2  {  public class Person  {  private string \_name;  private string \_address;  private string \_email;  public string Name  {  get { return \_name; }  set { \_name = value; }  }  public string Address  {  get { return \_address; }  set { \_address = value; }  }  public string Email  {  get { return \_address; }  set { \_address = value; }  }  }  } |
| using System;  using System.Collections.Generic;  namespace SEM\_cw2  {  public class Customer : Person  {  private List<Bike> \_bikes;  private int \_cardNumber;  private Boolean \_accepted;    public List<Bike> Bikes  {  get { return \_bikes; }  set { \_bikes = value; }  }  public int CardNumber  {  get { return \_cardNumber; }  set { \_cardNumber = value; }  }  public Boolean Accepted  {  get { return \_accepted; }  set { \_accepted = value; }  }    }  } |
| namespace SEM\_cw2  {  public class Worker : Person  {  private int \_staffNo;  private int \_jobID;  private double \_salary;  private int \_phoneNo;  public int StaffNumber  {  get { return \_staffNo; }  set { \_staffNo = value; }  }  public int JobID  {  get { return \_jobID; }  set { \_jobID = value; }  }  public double Salary  {  get { return \_salary; }  set { \_salary = value; }  }  public int PhoneNumber  {  get { return \_phoneNo; }  set { \_phoneNo = value; }  }    }  } |
| using System;  namespace SEM\_cw2  {  public class Bike  {  private int \_id;  private string \_type;  private string \_wheels;  private string \_size;  private string \_colour;  private string \_gears;  private string \_brakes;  private string \_handlebars;  private string \_saddle;  private Boolean \_extendedWarranty;  private double \_totalPrice;  private double \_estimatedDays;  private Customer \_customer;  public int ID  {  get { return \_id; }  set { \_id = value; }  }  public string Type  {  get { return \_type; }  set  {  if (value == "mountain" || value == "road")  {  \_type = value;  }  else  throw new ArgumentException("type is wrong");  }  }  public string Wheels  {  get { return \_wheels; }  set  {  if (value == "small" || value == "large")  {  \_wheels = value;  }  else  throw new ArgumentException("wheel type is wrong");  }  }  public string Size  {  get { return \_size; }  set  {  if (value == "small" || value == "medium" || value == "large")  {  \_size = value;  }  else  throw new ArgumentException("size is wrong");  }  }  public string Colour  {  get { return \_colour; }  set { \_colour = value; }  }  public string Gears  {  get { return \_gears; }  set  {  if (value == "Type A" || value == "Type B")  {  \_gears = value;  }  else  throw new ArgumentException("gear is wrong");  }  }  public string Brakes  {  get { return \_brakes; }  set  {  if (value == "Type A" || value == "Type B")  {  \_brakes = value;  }  else  throw new ArgumentException("brake type is wrong");  }  }  public string Handlebars  {  get { return \_handlebars; }  set  {  if (value == "Type A" || value == "Type B")  {  \_handlebars = value;  }  else  throw new ArgumentException("handlebar type is wrong");  }  }  public string Saddle  {  get { return \_saddle; }  set  {  if (value == "Type A" || value == "Type B")  {  \_saddle = value;  }  else  throw new ArgumentException("saddle type is wrong");  }  }  public Boolean ExtendedWarranty  {  get { return \_extendedWarranty; }  set { \_extendedWarranty = value; }  }  public double TotalPrice  {  get { return \_totalPrice; }  set { \_totalPrice = value; }  }  public double EstimatedDays  {  get { return \_estimatedDays; }  set { \_estimatedDays = value; }  }  public Customer Customer  {  get { return \_customer; }  set { \_customer = value; }  }  }  } |

## Test unit for Bike class:

|  |
| --- |
| using System;  using Microsoft.VisualStudio.TestTools.UnitTesting;  using SEM\_cw2;  namespace TestingClasses  {  [TestClass]  public class UnitTest1  {  //the bike object to test  Bike bike = new Bike();    [TestMethod]  [ExpectedException(typeof(ArgumentException))]  public void typeTest()  {  bike.Type = "hi";  }  [TestMethod]  public void typeTest2()  {  bike.Type = "small";  }  [TestMethod]  [ExpectedException(typeof(ArgumentException))]  public void wheelTest()  {  bike.Wheels = "asdasd";  }  [TestMethod]  [ExpectedException(typeof(ArgumentException))]  public void sizeTest()  {  bike.Wheels = "asdsad";  }  }  } |

## Results:

# User Interface

