Създайте Windows приложение **АВТОМОБИЛИ**, което съдържа:

- сортиран списък от автомобили;
- три празни текстови полета със съответни етикети пред тях **Име на модел**, **Обем на двигател** и **Скорост**;
- текстово поле за извеждане на резултатите;
- контрол за проверка коректността на въведените данни в текстовите полета;
- четири бутона:

Добави добавяне на данни за автомобил към наличния сортиран списък;

Промяна на скорост промяна скоростта на автомобил при зададено име на модел;

Списък изобразяване на сортирания списък от автомобили;

Изчисти изчистване на текстовите полета.

Създайте интерфейс **IVehicle**, представящ превозно средство.

1. Дефинирайте свойство за запис/четене **Speed**, което установява/връща скоростта на превозното средство.

Създайте клас **Car**, който наследява интерфейсите **IVehicle** и **IComparable**; съдържа информация за модел, обем на двигател и скорост.

1. Добавете конструктор с три параметъра.
2. Реализирайте свойството **Speed** на интерфейса **IVehicle**.
3. Реализирайте метода на интерфейса **IComparable** **int CompareTo (Object obj)**, така че да осигури бъдещо сортиране по обем на двигателя, при еднакви стойности на обема да сортира по скоростта, а при еднаква скорост – по модела.
4. Предефинирайте метода **public virtual string ToString()**, който връща символното представяне на класа.

В класа на формата:

1. Добавете поле за сортиран списък от автомобили (тип **SortedList**) и го инициализирайте.
2. Създайте метод за обработка на събитието **Click** за бутона **Добави**

- добавя автомобил към сортирания списък, като за ключ използва името на модела, а за стойност – самия автомобил; в полето за резултата изобразява данните за ново въведения автомобил;
- при наличие на автомобил с въведените данни в списъка в полето за резултата се извежда съобщение **ГРЕШКА!**;
- ако не са въведени данни в някое от текстовите полета, контролът за проверка на коректността на данните установява грешка за съответното текстово поле.

3. Създайте метод за обработка на събитието **Click** за бутона **Промяна на скорост**

- при наличие на въведени данни за модел на автомобил използвайте свойството **Speed**, за да промените скоростта на автомобила; полето за резултата извежда:
  обновените данни на намерения автомобил (при успешно търсене);
  Няма данни! (при неуспешно търсене);
- ако не са въведени данни за модел на автомобил, контролът за проверка на коректността на данните установява грешка за това текстово поле.

4. Създайте метод за обработка на събитието **Click** за бутона **Списък**

- извежда съдържанието на сортирания списък от автомобили в полето за резултата, като използва номератор **IDictionaryEnumerator**.

5. Създайте метод за обработка на събитието **Click** за бутона **Изчисти**

- изчиства всички текстови полета и контрола за проверка коректността на въведените данни.

1. Кои типове в Common Type System освобождават заетата памет чрез системата за събиране на боклука?

- ❑ всички типове;
- ❑ само референтните типове;
- ❑ всички типове без примитивните типове;
- ❑ само стойностните типове;

☒ само референтните типове

2. Кога е необходимо явно преобразуване (explicit casting)?

- ❑ при преобразуване от базов клас до клас наследник;
- ❑ при преобразуване от клас наследник до базов клас;
- ❑ и в двата случая.

☒ при преобразуване от базов клас до клас наследник

3. Кой от изброените начини за обхождане на масив бихте използвали, ако знаете, че елементите на масива трябва да се модифицират?

- ❑ оператор **for**;
- ❑ оператор **foreach**;
- ❑ обект от клас **IEnumerator**, получен с метод **GetEnumerator()**.

☒ оператор for

4. Декларирайте клас с име **MyClass** с **private** поле масив от 100 цели числа и индексатор, който хвърля изключение **InvalidOperationException** при излизане извън границите на масива.

```
class MyClass
{
   private int[] array = new int[100];
   public int this [int index]
   {
      get
      {  if (index<0 || index>=100)
            throw new InvalidOperationException("Индексът е извън областта");
         else return array[index];
      }
      set
      {  if (index<0 || index>=100)
            throw new InvalidOperationException("Индексът е извън областта");
         else array[index]=value;
      }
   }
}
```

5. Декларирайте интерфейс с име **IStorable** с два метода и едно свойство за запис/четене. Първият метод се нарича **Read**, няма параметри и връща целочислена стойност; вторият метод се нарича **Write**, има параметър от тип **Object** и не връща стойност. Свойството се нарича **Status** и е от тип **int**. Декларирайте клас с име **Document**, който реализира **IStorable**.

```
interface IStorable
{  int Read ();
   void Write (Object o);
   int Status {   get;   set;   }
}
public class Document : IStorable
{  private int status;
   public int Status
   {  get {   return status;   }
      set {   status = value;   }
   }
   public int Read ()   {  ...  }
   public void Write (Object o)   {   ...   }
}
```

6. Какъв ще бъде резултатът от изпълнението на следния код?

```
try
{  int num = 100;
   int den = 0;
   try
   {
        MessageBox.Show("Message1");
         int res = num/den;
         MessageBox.Show("Message2");
   }
   catch(ArithmeticException ae)
   {
         MessageBox.Show("Message3");
   }
   MessageBox.Show("Message4");
}
```

❑ Message1
Message4
Message1

❑ Message2
Message3
Message4

❑ Message1
Message3
Message4

❑ Message1
Message3
Message2
Message4

☒ Message1
Message3
Message4

7. Изключенията са:

- ❑ средство за спиране разпространението на грешки в програмния код;
- ❑ начин да се избегне проверката на кодове за състояние, връщани от функции;
- ❑ средство за възстановяване при грешка;
- ❑ обектно-ориентиран механизъм за обработка на грешки.

- ☒ средство за спиране разпространението на грешки в програмния код;
- ☒ начин да се избегне проверката на кодове за състояние, връщани от функции;
- ☒ обектно-ориентиран механизъм за обработка на грешки.

8. Дефинирайте:
   - клас с информация за събитието **Pick**;
   - **public** делегат за обработка на събитието **Pick**;
   - **public** събитие **Pick**;
   - манипулатор на събитието **Pick**.

```
class PickEventArgs : EventArgs
{…}
public delegate void PickEventArgsHandler (object source,
                                           PickEventArgs e);
public event PickEventArgsHandler Pick;
void OnPick (object source, PickEventArgs e)
{…}
```

9. В кода за обработка на кое събитие за формата бихте включили инициализиращ код за контролите?

- ❑ Activated
- ❑ Load
- ❑ Closed

☒ Load

10. Кое от следните събития ще използвате, за да направите проверка за коректно въведени данни?

- ❑ LostFocus
- ❑ Validating
- ❑ Leave
- ❑ Validated

☒ Validating