```
using System;

// Interface 1: Các chức năng cơ bản của phương tiện
public interface IVehicle
{
    void Start();
    void Stop();
    double GetFuelLevel();
}

// Interface 2: Các chức năng định vị và dẫn đường
public interface INavigable
{
    void SetDestination(string destination);
    void Navigate();
    string GetCurrentLocation();
}

// Interface 3: Các chức năng kết nối
public interface IConnectable
{
    void ConnectBluetooth();
    void ConnectWiFi();
    void DisconnectAll();
}

// Interface 4: Các chức năng bảo hiểm
public interface IInsurable
{
    string GetInsuranceInfo();
    void RenewInsurance();
    void ClaimInsurance();
}

// Class cha abstract
public abstract class Vehicle
{
    public string Brand { get; set; }
    public string Model { get; set; }
    public int Year { get; set; }
    public double FuelLevel { get; protected set; }

    public Vehicle(string brand, string model, int year)
    {
        Brand = brand;
        Model = model;
        Year = year;
        FuelLevel = 50.0; // Mức nhiên liệu mặc định
    }
```

```
50
51      public virtual void DisplayInfo()
52      {
53          Console.WriteLine($"Vehicle: {Brand} {Model} ({Year})");
54          Console.WriteLine($"Fuel Level: {FuelLevel:F1} liters");
55      }
56
57      public void AddFuel(double amount)
58      {
59          FuelLevel += amount;
60          Console.WriteLine($"Đã thêm {amount} lít nhiên liệu");
61      }
62
63      // Phương thức abstract bắt buộc phải override
64      public abstract void Move();
65  }
66
67  // Class Car - hiện thực tất cả 4 interface (xe hơi cao cấp)
68  public class Car : Vehicle, IVehicle, INavigable, IConnectable, IInsurable
69  {
70      public int NumberOfDoors { get; set; }
71      public bool HasAirCondition { get; set; }
72      private string currentLocation = "Garage";
73      private string destination = "";
74
75      public Car(string brand, string model, int year, int doors, bool
          hasAC)
76          : base(brand, model, year)
77      {
78          NumberOfDoors = doors;
79          HasAirCondition = hasAC;
80      }
81
82      public override void Move()
83      {
84          Console.WriteLine($"{Brand} {Model} đang di chuyển êm ái trên
              đường");
85      }
86
87      public override void DisplayInfo()
88      {
89          base.DisplayInfo();
90          Console.WriteLine($"Số cửa: {NumberOfDoors}");
91          Console.WriteLine($"Có điều hòa: {(HasAirCondition ? "Có" :
              "Không")}");
92      }
93
94      // Hiện thực IVehicle
95      public void Start()
```

```
 96      {
 97          Console.WriteLine($"{Brand} {Model} khởi động êm ru");
 98          FuelLevel -= 0.1;
 99      }
100
101      public void Stop()
102      {
103          Console.WriteLine($"{Brand} {Model} dừng lại an toàn");
104      }
105
106      public double GetFuelLevel()
107      {
108          return FuelLevel;
109      }
110
111      // Hiện thực INavigable
112      public void SetDestination(string destination)
113      {
114          this.destination = destination;
115          Console.WriteLine($"Đã đặt đích đến: {destination}");
116      }
117
118      public void Navigate()
119      {
120          if (!string.IsNullOrEmpty(destination))
121          {
122              Console.WriteLine($"GPS đang dẫn đường đến {destination}");
123              currentLocation = destination;
124          }
125      }
126
127      public string GetCurrentLocation()
128      {
129          return currentLocation;
130      }
131
132      // Hiện thực IConnectable
133      public void ConnectBluetooth()
134      {
135          Console.WriteLine("Đã kết nối Bluetooth với điện thoại");
136      }
137
138      public void ConnectWiFi()
139      {
140          Console.WriteLine("Đã kết nối WiFi hotspot");
141      }
142
143      public void DisconnectAll()
144      {
```

```
145         Console.WriteLine("Đã ngắt tất cả kết nối");
146     }
147
148     // Hiện thực IInsurable
149     public string GetInsuranceInfo()
150     {
151         return $"Bảo hiểm xe {Brand} {Model} - Hạn đến: 31/12/2025";
152     }
153
154     public void RenewInsurance()
155     {
156         Console.WriteLine("Đã gia hạn bảo hiểm xe ô tô");
157     }
158
159     public void ClaimInsurance()
160     {
161         Console.WriteLine("Đã gửi yêu cầu bồi thường bảo hiểm");
162     }
163 }
164
165 // Class Motorcycle - hiện thực 3 interface (không có connectivity)
166 public class Motorcycle : Vehicle, IVehicle, INavigable, IInsurable
167 {
168     public bool HasSidecar { get; set; }
169     public int EngineSize { get; set; }
170     private string currentLocation = "Parking";
171     private string destination = "";
172
173     public Motorcycle(string brand, string model, int year, bool
        hasSidecar, int engineSize)
174         : base(brand, model, year)
175     {
176         HasSidecar = hasSidecar;
177         EngineSize = engineSize;
178     }
179
180     public override void Move()
181     {
182         Console.WriteLine($"{Brand} {Model} đang phóng nhanh trên đường");
183     }
184
185     public override void DisplayInfo()
186     {
187         base.DisplayInfo();
188         Console.WriteLine($"Có xe phụ: {(HasSidecar ? "Có" : "Không")}");
189         Console.WriteLine($"Dung tích động cơ: {EngineSize}cc");
190     }
191
192     // Hiện thực IVehicle
```

```
193     public void Start()
194     {
195         Console.WriteLine($"{Brand} {Model} nổ máy ầm ầm");
196         FuelLevel -= 0.05;
197     }
198
199     public void Stop()
200     {
201         Console.WriteLine($"{Brand} {Model} dừng lại");
202     }
203
204     public double GetFuelLevel()
205     {
206         return FuelLevel;
207     }
208
209     // Hiện thực INavigable
210     public void SetDestination(string destination)
211     {
212         this.destination = destination;
213         Console.WriteLine($"Đã nhớ đích đến: {destination}");
214     }
215
216     public void Navigate()
217     {
218         if (!string.IsNullOrEmpty(destination))
219         {
220             Console.WriteLine($"Đang đi đến {destination} theo kinh
                  nghiệm");
221             currentLocation = destination;
222         }
223     }
224
225     public string GetCurrentLocation()
226     {
227         return currentLocation;
228     }
229
230     // Hiện thực IInsurable
231     public string GetInsuranceInfo()
232     {
233         return $"Bảo hiểm xe máy {Brand} {Model} - Loại cơ bản";
234     }
235
236     public void RenewInsurance()
237     {
238         Console.WriteLine("Đã gia hạn bảo hiểm xe máy");
239     }
240
```

```
241     public void ClaimInsurance()
242     {
243         Console.WriteLine("Đã gửi đơn bồi thường tai nạn xe máy");
244     }
245 }
246 
247 // Class Truck - hiện thực 2 interface (xe tải cơ bản)
248 public class Truck : Vehicle, IVehicle, IConnectable
249 {
250     public double LoadCapacity { get; set; }
251     public double CurrentLoad { get; private set; }
252 
253     public Truck(string brand, string model, int year, double capacity)
254         : base(brand, model, year)
255     {
256         LoadCapacity = capacity;
257         CurrentLoad = 0;
258     }
259 
260     public override void Move()
261     {
262         Console.WriteLine($"{Brand} {Model} đang chở hàng di chuyển chậm
              rãi");
263     }
264 
265     public override void DisplayInfo()
266     {
267         base.DisplayInfo();
268         Console.WriteLine($"Tải trọng tối đa: {LoadCapacity} tấn");
269         Console.WriteLine($"Hàng hóa hiện tại: {CurrentLoad} tấn");
270     }
271 
272     // Hiện thực IVehicle
273     public void Start()
274     {
275         Console.WriteLine($"{Brand} {Model} khởi động động cơ diesel");
276         FuelLevel -= 0.2;
277     }
278 
279     public void Stop()
280     {
281         Console.WriteLine($"{Brand} {Model} dừng lại với tiếng phanh
              hơi");
282     }
283 
284     public double GetFuelLevel()
285     {
286         return FuelLevel;
287     }
```

```
288
289     // Hiện thực IConnectable
290     public void ConnectBluetooth()
291     {
292         Console.WriteLine("Đã kết nối Bluetooth với hệ thống quản lý đội
              xe");
293     }
294
295     public void ConnectWiFi()
296     {
297         Console.WriteLine("Đã kết nối WiFi để theo dõi GPS");
298     }
299
300     public void DisconnectAll()
301     {
302         Console.WriteLine("Đã ngắt kết nối tất cả thiết bị");
303     }
304
305     // Phương thức riêng cho xe tải
306     public void LoadCargo(double weight)
307     {
308         if (CurrentLoad + weight <= LoadCapacity)
309         {
310             CurrentLoad += weight;
311             Console.WriteLine($"Đã chất {weight} tấn hàng. Tổng:
                  {CurrentLoad} tấn");
312         }
313         else
314         {
315             Console.WriteLine("Không thể chất thêm hàng - vượt quá tải
                  trọng!");
316         }
317     }
318
319     public void UnloadCargo()
320     {
321         CurrentLoad = 0;
322         Console.WriteLine("Đã dỡ hết hàng hóa");
323     }
324 }
325
326 // Chương trình chính
327 class Program
328 {
329     static void Main()
330     {
331         Console.WriteLine("=== VÍ DỤ KẾ THỪA VÀ NHIỀU INTERFACE TRONG C#
              ===\n");
332
```

```
333         // Tạo xe ô tô (có tất cả tính năng)
334         Car luxuryCar = new Car("Mercedes", "S-Class", 2023, 4, true);
335         Console.WriteLine("--- XE Ô TÔ CAO CẤP ---");
336         luxuryCar.DisplayInfo();
337         luxuryCar.Start();
338         luxuryCar.ConnectBluetooth();
339         luxuryCar.SetDestination("Sân bay Tân Sơn Nhất");
340         luxuryCar.Navigate();
341         Console.WriteLine($"Vị trí hiện tại: {luxuryCar.GetCurrentLocation
              ()}");
342         Console.WriteLine(luxuryCar.GetInsuranceInfo());
343         Console.WriteLine();
344
345         // Tạo xe máy (không có connectivity)
346         Motorcycle sportBike = new Motorcycle("Yamaha", "R1", 2022,
              false, 1000);
347         Console.WriteLine("--- XE MÁY THỂ THAO ---");
348         sportBike.DisplayInfo();
349         sportBike.Start();
350         sportBike.SetDestination("Quận 1");
351         sportBike.Navigate();
352         Console.WriteLine(sportBike.GetInsuranceInfo());
353         Console.WriteLine();
354
355         // Tạo xe tải (chỉ có chức năng cơ bản)
356         Truck deliveryTruck = new Truck("Isuzu", "NPR", 2023, 5.0);
357         Console.WriteLine("--- XE TẢI GIAO HÀNG ---");
358         deliveryTruck.DisplayInfo();
359         deliveryTruck.LoadCargo(2.5);
360         deliveryTruck.Start();
361         deliveryTruck.ConnectWiFi();
362         deliveryTruck.Move();
363         deliveryTruck.UnloadCargo();
364         Console.WriteLine();
365
366         // Sử dụng Interface Polymorphism
367         Console.WriteLine("--- SỬ DỤNG INTERFACE POLYMORPHISM ---");
368
369         // Tất cả đều implement IVehicle
370         IVehicle[] vehicles = { luxuryCar, sportBike, deliveryTruck };
371         Console.WriteLine("Khởi động tất cả phương tiện:");
372         foreach (IVehicle vehicle in vehicles)
373         {
374             vehicle.Start();
375         }
376         Console.WriteLine();
377
378         // Chỉ xe có thể navigate
379         INavigable[] navigableVehicles = { luxuryCar, sportBike };
```

```
380         Console.WriteLine("Dẫn đường cho các phương tiện có GPS:");
381         foreach (INavigable nav in navigableVehicles)
382         {
383             nav.SetDestination("Bến Thành Market");
384             nav.Navigate();
385         }
386         Console.WriteLine();
387 
388         // Chỉ xe có thể kết nối
389         IConnectable[] connectableVehicles = { luxuryCar, deliveryTruck };
390         Console.WriteLine("Kết nối các thiết bị thông minh:");
391         foreach (IConnectable conn in connectableVehicles)
392         {
393             conn.ConnectBluetooth();
394         }
395     }
396 }
```