OOP Sample HL

To improve customer satisfaction, a supermarket chain wants to create an object-oriented program (OOP) to simulate the lines of customers at the check-outs in their point of sale (POS) system.

This point of sale (POS) system consists of several check-out counters. After filling their shopping carts with items, customers line up at one of the check-out counters. In most cases, they wait in line until it is their turn to pay.

Three real-world objects are implemented using the following classes:

`Cart`	represents a customer together with the items they intend to purchase
`POSline`	represents an individual check-out counter and the line of customers with carts waiting at that checkout
`POSsystem`	the overall check-out system

The UML diagram for the class `POSline` is provided below.

`POSline`	

`- String id`
`- boolean active`
`- Cart[] line`	

`+ constructor`	
`+ Cart getLine(int n)`	- returns the cart object at position `n`
`… more accessor/mutator methods`	
`+ void joinLine(Cart newCart)`	- adds a new cart object to the end of the line
`+ void checkoutCart()`	- removes the first cart object in the line
`+ Cart leaveLine(int n)`	- removes and returns the cart object at position `n`
`… more methods`	

1. (a) State the relationship between the `POSsystem` and `POSline` objects. [1]

   (b) Draw a diagram to show the relationship between the objects `POSsystem`, `POSline` and `Cart`. You are not required to draw a complete UML diagram. [2]

   (c) Define the term *identifier*. [1]

   (d) Distinguish between a class and an instantiation. You must make reference to the UML provided. [3]

(e) State the code fragment that instantiates an array `line` of 20 `Cart` objects. [2]

(f) Construct the method `public void joinLine(Cart newCart)` that adds a `newCart` to `line` in the next empty `array` location. You may assume that `line` is not full. [3]

(g) Construct the method `public Cart leaveLine(int n)` that removes the cart at position `n` from `line` by shifting down all the array entries bigger than `n`, before returning the indicated `Cart` object. You may assume that a cart exists at position `n`. [4]

The supermarket chain wants to use this OOP simulation to experiment with different ways of organizing their check-out system. For example, it is possible to have different check-out counters such as cash-only or card only, or to have check-out counters for ten items or fewer.

2. (a) Define the term *inheritance*. [2]

(b) Explain **one** advantage of the OOP feature "inheritance" with reference to this scenario. [3]

(c) Describe **two** advantages of using libraries of classes. [4]

It is also possible to have one line that serves a number of check-out counters.

(d) Outline **one** change needed to the original model to allow this possibility to be implemented. [2]

(e) Outline **two** reasons why the use of multiple programming teams in different locations may be problematic when developing an integrated software solution. [4]

3. When the number of customers in the supermarket is low, some check-out counters will close. When there are many customers waiting, a new check-out counter will open.

(a) Outline why the variable `active` in the UML of the class `POSline` was defined as a boolean data type. [2]

When a new check-out counter opens, some customers from the nearest line will choose to move their carts to this check-out counter.

For this simulation, the assumption is made that every second cart in the old line will move to the new line and the other carts will remain in the original line.

(b) Construct the code for a method `split` (part of the class `POSsystem`), that takes an existing, non-empty `POSline` as a parameter. It should copy every second cart from the original line into a new line. Afterwards it should delete every second cart from the original line.

An example call in `POSsystem` would be: `POSline number2 = split(number1)`, where `number1` is an existing `POSline`.

You may use any method declared or developed. [8]

As a result of the simulation, the company has decided to create a new POS system for their supermarkets.

There have been discussions about adapting existing open source code when developing modules of the new POS system.

(c) Describe **two** ethical issues that may arise if modules of the new POS system are developed from open source code. [4]

4. The array `line` in `POSline` is now replaced by a singly linked list `list` of `CartNode` objects. The class `CartNode` has been defined as follows.

```
public class CartNode
{
   private Cart myCart;
   private CartNode next;

   public CartNode(Cart aCart)
   {
      this.myCart = aCart;
      this.next = null;
   }
   public Cart getCart(){ return this.myCart; }
   public CartNode getNext(){ return this.next; }
   public void setNext(CartNode nextNode)
   {
      this.next = nextNode;
   }
}
```

(a) Define the term *object reference*. [2]

The class `POSlist` has been implemented with the standard list methods `addLast` and `removeFirst` that act on `list`.

(b) Using object references, construct the method `public Cart removeFirst()` that removes the first `CartNode` object from `list`. The method must return the `Cart` object in that node or `null` if `list` is empty. [4]

(c) Sketch the linked list `list` after running the following code fragment where `cart1`, `cart2`, `cart3`, `cart4` and `cart5` have been instantiated as objects of the class `Cart`.

```
POSlist queueList = new POSlist()
queueList.addLast(cart2);
queueList.addLast(cart1);
queueList.addLast(cart4);
queueList.removeFirst();
queueList.addLast(cart5);
queueList.addLast(cart3);
queueList.removeFirst();
```

[2]

(d) Outline **one** feature of the abstract data structure *queue* that makes it unsuitable to implement customers waiting in line. [2]

A method `leaveList(int n)` is required in the class `POSlist`, similar to the method `leaveLine(int n)` that was added to the class `POSline`.

(e) Using object references, construct the method `public Cart leaveList(int n)` that removes the nth `CartNode` object from `list`. The method must return the `Cart` object in that node.

You may assume that the nth `CartNode` object exists in the list.
You may use any method declared or developed. [6]

(f) Explain the importance of using coding style and naming conventions when programming. [4]