1. Problem Scenario

In a large family, there are many uncles (and aunts) and many nieces (and nephews). Each uncle buys a present for each niece on her birthday every year. Unfortunately, this can cause all sorts of problems:

- Last year Amy received a charming book ('The Wonder of Computers') from Uncle Albert. Unfortunately, Uncle Bill gave it to her too.
- When Beatrice opened her present from Uncle Charlie, she found it was the same thing he had given to Claire. "How could he?" she shouted rudely, "I'm not at all like Claire!".
- Uncle David is getting forgetful. Little Emily was so upset not to get a present.

1.1 Program Requirements

The family decide they need a computer program to manage the giving of presents. The requirements are:

1. Uncles and nieces can be added to the system. The date of each niece's birthday is recorded.
2. A list of uncles can be generated – in alphabetical order by name.
3. A list of nieces can be generated – in order of birthday.
4. The system holds a list of the presents selected by each uncle for the next birthday of one of his nieces. Each present is described in a few words.
5. An uncle can enter the present he intends to give to one of his nieces. The program ensures that:
    i. each niece receives something different from each uncle
    ii. each uncles gives something different to each niece.
6. A list of the presents given by one of the uncles can be generated, showing the niece who is to receive it. The nieces for whom no present has been chosen should also be listed.
7. A list of presents to be received by one of the nieces can be generated, showing the uncle giving it. The uncles who have no present for the niece should also be listed.
8. The list of presents for a niece can be deleted (that's done when her birthday is past).

1.2 Your Task

Write a program to meet the requirements. To make a start, write classes to provide the behaviour, without a user interface. The public classes you should provide are: Family, Niece and Uncle. The public methods of these classes create an interface (often called an API – Application Program Interface) to which a user interface could be attached. The API is described below; note that your program will also contain other classes and methods.

Class Family

Class representing a family.

Constructor Summary	
`Family()` Create an empty family with no uncles, nieces or presents.	

Method Summary	
boolean	**addNiece**(java.lang.String name, int day, int month) Add a new niece. If there is already a niece of this name, false is returned and nothing is added.
boolean	**addUncle**(java.lang.String name) Add a new uncle. If there is already an uncle of the name, false is returned and nothing is added.
Niece	**findNiece**(java.lang.String name) Lookup a niece by name; return null if not found.
Uncle	**findUncle**(java.lang.String name) Lookup an uncle by name; return null if not found.
void	**listNieces**() List (to the console) the nieces recorded.
void	**listUncles**() List (to the console) the uncles recorded.

Class Uncle

Class representing an uncle. Note that the constructor is not public – use Family.addUncle().

Method Summary	
boolean	**addPresent**(Niece recipient, java.lang.String description) Adds a new present, given by this uncle. Return true if the present is allowed.
void	**listPresents**() Lists (to the console) the presents given by this uncle, showing the recipient. Nieces with no present from this uncle are also listed.

Class Niece

Class representing a niece. Note that the constructor is not public – use Family.addNiece().

Method Summary	
int	**clearPresents**() Delete all the presents chosen for this niece. Return the number removed.
void	**listPresents**() Lists (to the console) the presents to be received by this niece, showing the giver. Uncles with no present for this niece are also listed.