Christmas

Story: Christmas is coming and a lot of children and adults are going to buy toys. Both the children and adults have some things in common. For this example consider them having a name, age and an int value which holds the budget they have for shopping. The mentioned attributes are **common** so place them accordingly in a parent class. The customers in our shop can buy toys. Every customer can buy a toy if they have the money that toy costs. However, take into account that children under the age of 14 cannot buy toys (use `@Override`). Simulate a shopping day: create some customers and toys and make them spend some money. (Place your `main` method in `Shop` class).

Christmas

Story: Christmas is coming and a lot of children and adults are going to buy toys. Both the children and adults have some things in common. For this example consider them having a name, age and an int value which holds the budget they have for shopping. The mentioned attributes are **common** so place them accordingly in a parent class. The customers in our shop can buy toys. Every customer can buy a toy if they have the money that toy costs. However, take into account that children under the age of 14 cannot buy toys (use `@Override`). Simulate a shopping day: create some customers and toys and make them spend some money. (Place your `main` method in `Shop` class).

Christmas

Story: Christmas is coming and a lot of children and adults are going to buy toys. Both the children and adults have some things in common. For this example consider them having a name, age and an int value which holds the budget they have for shopping. The mentioned attributes are **common** so place them accordingly in a parent class. The customers in our shop can buy toys. Every customer can buy a toy if they have the money that toy costs. However, take into account that children under the age of 14 cannot buy toys (use `@Override`). Simulate a shopping day: create some customers and toys and make them spend some money. (Place your `main` method in `Shop` class).

1. Read and understand the story. Implement the story in Java code.
2. Implement the ability of sorting an array of customers by their budget and exemplify this.
3. Loop through your array of customers and only print the names of children.
4. Add a new attribute in your toy class: `type`. This type should be able to be set to the next three values: *car, doll* and *educational* (Use constants).
5. Add a GUID to your toys which is generated upon creation. Add `getDetails():String` in your toy class. Inside this method check the type of our toy. Return the GUID if its type is *car*, a message if *doll* and its price if *educational* type. (Use constants)
6. Add two interfaces for our toys (`ElectronicToy_I` with `+consumePower():void` and `MultiPiecesToy_I` with `+countPieces():void`). Create two new types of toys that implement the created interfaces.
7. Create the class diagram of the code you have written.

1. Read and understand the story. Implement the story in Java code.
2. Implement the ability of sorting an array of customers by their budget and exemplify this.
3. Loop through your array of customers and only print the names of children.
4. Add a new attribute in your toy class: `type`. This type should be able to be set to the next three values: *car, doll* and *educational* (Use constants).
5. Add a GUID to your toys which is generated upon creation. Add `getDetails():String` in your toy class. Inside this method check the type of our toy. Return the GUID if its type is *car*, a message if *doll* and its price if *educational* type. (Use constants)
6. Add two interfaces for our toys (`ElectronicToy_I` with `+consumePower():void` and `MultiPiecesToy_I` with `+countPieces():void`). Create two new types of toys that implement the created interfaces.
7. Create the class diagram of the code you have written.

1. Read and understand the story. Implement the story in Java code.
2. Implement the ability of sorting an array of customers by their budget and exemplify this.
3. Loop through your array of customers and only print the names of children.
4. Add a new attribute in your toy class: `type`. This type should be able to be set to the next three values: *car, doll* and *educational* (Use constants).
5. Add a GUID to your toys which is generated upon creation. Add `getDetails():String` in your toy class. Inside this method check the type of our toy. Return the GUID if its type is *car*, a message if *doll* and its price if *educational* type. (Use constants)
6. Add two interfaces for our toys (`ElectronicToy_I` with `+consumePower():void` and `MultiPiecesToy_I` with `+countPieces():void`). Create two new types of toys that implement the created interfaces.
7. Create the class diagram of the code you have written.