Westworld

Story: Westworld is a fictional world in which two types of creatures live: hosts and guests. Both the hosts and guests have some things in common. For this example consider them having a name, age and a boolean value which says if they're dead or not. The mentioned attributes are **common** so place them accordingly in a parent class. The creatures in our world can shoot other creatures. Every creature can die as well, but keep in mind we are in Westworld and guests must not die(use `@Override`). Simulate a shootout in our world: create some hosts and guests and shoot at random (Place your `main` method in `Westworld` class).

Westworld

Story: Westworld is a fictional world in which two types of creatures live: hosts and guests. Both the hosts and guests have some things in common. For this example consider them having a name, age and a boolean value which says if they're dead or not. The mentioned attributes are **common** so place them accordingly in a parent class. The creatures in our world can shoot other creatures. Every creature can die as well, but keep in mind we are in Westworld and guests must not die(use `@Override`). Simulate a shootout in our world: create some hosts and guests and shoot at random (Place your `main` method in `Westworld` class).

Westworld

Story: Westworld is a fictional world in which two types of creatures live: hosts and guests. Both the hosts and guests have some things in common. For this example consider them having a name, age and a boolean value which says if they're dead or not. The mentioned attributes are **common** so place them accordingly in a parent class. The creatures in our world can shoot other creatures. Every creature can die as well, but keep in mind we are in Westworld and guests must not die(use `@Override`). Simulate a shootout in our world: create some hosts and guests and shoot at random (Place your `main` method in `Westworld` class).

1. Read and understand the story. Implement the story in Java code.
2. Implement the ability of sorting an array of creatures by their name and exemplify this.
3. Loop through your array of creatures and only print the names of hosts.
4. Add a new attribute to your host: `mode`. This mode should be able to be set to the next three values: *normal, analysis* and *shutdown* (Use constants).
5. Add a GUID to your hosts which is generated upon creation. Add `getNextRemark():String` in your host. Inside this method check the mode of our host. Return the GUID if in *analysis*, a message if *shutdown* and their name and age if *normal* mode. (Use constants)
6. Add two interfaces for our hosts (`CowboyHost_I` with `+catchVillans():void` and `DrunkHost_I` with `+drink():void`). Create two new types of hosts 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 creatures by their name and exemplify this.
3. Loop through your array of creatures and only print the names of hosts.
4. Add a new attribute to your host: `mode`. This mode should be able to be set to the next three values: *normal, analysis* and *shutdown* (Use constants).
5. Add a GUID to your hosts which is generated upon creation. Add `getNextRemark():String` in your host. Inside this method check the mode of our host. Return the GUID if in *analysis*, a message if *shutdown* and their name and age if *normal* mode. (Use constants)
6. Add two interfaces for our hosts (`CowboyHost_I` with `+catchVillans():void` and `DrunkHost_I` with `+drink():void`). Create two new types of hosts 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 creatures by their name and exemplify this.
3. Loop through your array of creatures and only print the names of hosts.
4. Add a new attribute to your host: `mode`. This mode should be able to be set to the next three values: *normal, analysis* and *shutdown* (Use constants).
5. Add a GUID to your hosts which is generated upon creation. Add `getNextRemark():String` in your host. Inside this method check the mode of our host. Return the GUID if in *analysis*, a message if *shutdown* and their name and age if *normal* mode. (Use constants)
6. Add two interfaces for our hosts (`CowboyHost_I` with `+catchVillans():void` and `DrunkHost_I` with `+drink():void`). Create two new types of hosts that implement the created interfaces.
7. Create the class diagram of the code you have written.