**FINAL CODING EXAM**

**1. `Transportation` class (30 Point):**

**- The `Transportation` class is an abstract class that represents a general form of transportation.**

**- It contains four properties: `name`, `destination`, `speed`, and `passengers`.**

**- There are two constructors: one with no arguments, and one with `name`, `destination`, and `speed` arguments.**

**- There are accessor methods for `name`, `destination`, `speed`, and `passengers`.**

**- There is a mutator method to add passengers to the `passengers` ArrayList.**

**- It also contains two abstract methods: `totalPeople()` and `toString()`. (10)**

**- The `totalPeople()` method returns the total number of people on the transportation.**

**- The `toString()` method returns a String representation of the `Transportation` object.**

**2. `Passport` interface (10 Point):**

**- The `Passport` interface represents a passport that can be checked to determine if it is valid for travel.**

**- It contains one abstract method: `passportCheck()`.**

**- The `passportCheck()` method returns a boolean value indicating whether or not the passport is valid.**

**3. `Airplane` class (20 Point):**

**- The `Airplane` class extends the `Transportation` class and implements the `Passport` interface.**

**- It contains two additional properties: `pilots` and `belongings`.**

**- There are two constructors: one with no arguments, and one with `name`, `destination`, and `speed` arguments.**

**- There are accessor methods for `pilots` and `belongings`.**

**- There are mutator methods to add pilots and belongings to the respective ArrayLists.**

**- It overrides the `totalPeople()` method from the `Transportation` class to include the number of pilots.**

**- It also overrides the `toString()` method from the `Transportation` class to include the airline name, speed, and total number of people.**

**4. `SchoolBus` class (20 Point):**

**- The `SchoolBus` class extends the `Transportation` class.**

**- It contains one additional property: `drivers`.**

**- There are two constructors: one with no arguments, and one with `name`, `destination`, and `speed` arguments.**

**- There is an accessor method for `drivers`.**

**- There is a mutator method to add drivers to the `drivers` ArrayList.**

**- It overrides the `totalPeople()` method from the `Transportation` class to include the number of drivers.**

**- It also overrides the `toString()` method from the `Transportation` class to include the school bus name, speed, and total number of people.**

**5. `Display` class (20 Point):**

**- The `Display` class contains two methods to display the passengers on either an `Airplane` or `SchoolBus`.**

**- There is a constructor with no arguments.**

**- The `airplanePassengers()` method takes a `Transportation` object as a parameter and displays the passengers on the `Airplane`.**

**- The `schoolBusPassengers()` method takes a `Transportation` object as a parameter and displays the passengers on the `SchoolBus`.**

**- Both methods use the `instanceof` keyword to check if the `Transportation` object is an `Airplane` or a `SchoolBus`, and then display the appropriate passengers using a `for` loop.**

**DETAIL INFORMATION**

**## Transportation**

**### Properties**

**- `private String name`: name of the transportation.**

**- `private String destination`: destination of the transportation.**

**- `private double speed`: speed of the transportation.**

**- `private final ArrayList<String> passengers`: an ArrayList of passengers that are in the transportation.**

**### Constructors**

**- `Transportation()`: creates an instance of the Transportation class with an empty ArrayList of passengers.**

**- `Transportation(String name, String destination, double speed)`: creates an instance of the Transportation class with a name, a destination, a speed, and an empty ArrayList of passengers.**

**### Accessors and Mutators**

**- `public String getName()`: returns the name of the transportation.**

**- `public String getDestination()`: returns the destination of the transportation.**

**- `public double getSpeed()`: returns the speed of the transportation.**

**- `public ArrayList<String> getPassengers()`: returns the ArrayList of passengers in the transportation.**

**- `public void setPassengers(String passengers)`: adds a passenger to the ArrayList of passengers in the transportation.**

**### Abstract Methods**

**- `public abstract int totalPeople()`: returns the total number of people in the transportation.**

**- `public abstract String toString()`: returns a String representation of the transportation.**

**## Passport**

**### Methods**

**- `public abstract boolean passportCheck()`: checks if a passport is present.**

**## Airplane**

**### Properties**

**- `private final ArrayList<String> pilots`: an ArrayList of pilots that are in the airplane.**

**- `private final ArrayList<String> belongings`: an ArrayList of belongings that are in the airplane.**

**### Constructors**

**- `Airplane()`: creates an instance of the Airplane class with an empty ArrayList of pilots and belongings.**

**- `Airplane(String name, String destination, double speed)`: creates an instance of the Airplane class with a name, a destination, a speed, and an empty ArrayList of pilots and belongings.**

**### Accessors and Mutators**

**- `public ArrayList<String> getPilots()`: returns the ArrayList of pilots in the airplane.**

**- `public ArrayList<String> getBelongings()`: returns the ArrayList of belongings in the airplane.**

**- `public void setPilots(String pilots)`: adds a pilot to the ArrayList of pilots in the airplane.**

**- `public void setBelongings(String belongings)`: adds a belonging to the ArrayList of belongings in the airplane.**

**### Implemented Method**

**- `public boolean passportCheck()`: checks if a passport is present in the belongings ArrayList (Check whether String “Passport” is inside of the belongings ArrayList).**

**### Inherited Methods**

**- `public int totalPeople()`: returns the total number of people in the airplane.**

**- `public String toString()`: returns a String representation of the airplane.**

**## SchoolBus**

**### Properties**

**- `private final ArrayList<String> drivers`: an ArrayList of drivers that are in the school bus.**

**### Constructors**

**- `SchoolBus()`: creates an instance of the SchoolBus class with an empty ArrayList of drivers.**

**- `SchoolBus(String name, String destination, double speed)`: creates an instance of the SchoolBus class with a name, a destination, a speed, and an empty ArrayList of drivers.**

**### Accessors and Mutators**

**- `public ArrayList<String> getDrivers()`: returns the ArrayList of drivers in the school bus.**

**- `public void setDrivers(String drivers)`: adds a driver to the ArrayList of drivers in the school bus.**

**### Inherited Methods**

**- `public int totalPeople()`: returns the total number of people in the school bus.**

**- `public String toString()`: returns a String representation of the school bus.**

**## Display**

**### Constructors**

**- `Display()`: creates an instance of the Display class.**

**### Methods**

**- `airplanePassengers(Transportation transportation)`: takes a `Transportation` object as input and prints the passengers on the airplane (if the input is an instance of the `Airplane` class).**

**- `schoolBusPassengers(Transportation transportation)`: takes a `Transportation` object as input and prints the passengers on the school bus (if the input is an instance of the `SchoolBus` class).**

**TEST CODE OUTPUT**

**(You should have same result as following)**

**A screenshot of a computer

Description automatically generated with medium confidence**