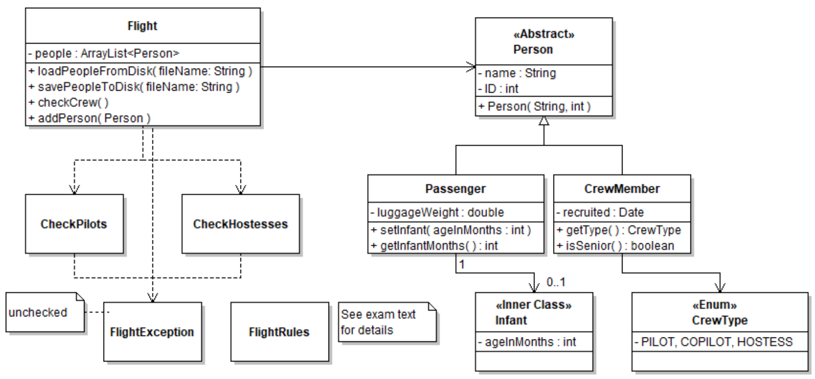
BLM2562 OBJECT ORIENTED PROG., Section 1&2, 2nd MIDTERM **(10/05/2018)**

Duration:	90mins.	**Score:**					**Student Nr:**	**Signature:**
Grading:	**1**	**2**	**3**	**4**	**5**	**6**	**Name, Surname:**	
	10	15	15	10	15	35		

QUESTIONS

Açıklamalı [YES1]: CheckCrew metoduna benzer checkFlight yapıp yolcu üzerinden de kontrol yapılacak şekilde soru genişletilebilir.



```java
public class FlightRules {
 public final static int minHostessCount = 2;
 public final static int minSeniorHostessCount = 1;
 public final static int yearsToBecomeSenior = 3;
}
```

Answer these questions according to the UML class schema and the code given above. You may need to extract hidden information from the schema and add necessary code while answering the questions.

Question 1: Write the source code of enum CrewType.

Question 2: Write the source code of the class CrewMember. A crew member cannot change its duty type. For example, a hostess can never become a pilot. The number of years to become a senior member should be obtained from the class FlightRules.

Question 3: Write the source code of the class Passenger. A passenger can travel with an infant (i.e. his/her child). Code the class Infant as an inner class of the class Passenger. If a passenger is not travelling with an infant, the getInfantMonths method must return –1.

Question 4: Write the source code of the exception FlightException.

Question 5: Write the source code of the multithreaded class of CheckPilots. The details of this class is given in Question 6 as an instance of this class is used in the Flight.checkCrew() method.

Question 6: Write the source code of the class Flight. Some details of this class is as follows:

- addPerson: This method creates a FlightException if a person with the same ID is attempted to add to the people list multiple times.
- checkCrew: This method is responsible from creating an instance of CheckHostesses and an instance of CheckPilots in separate threads and from executing them. The CheckHostesses instance checks the people list for hostesses. A flight must contain at least two hostesses and at least one of them must be senior worker. The CheckPilots instance checks the people list for pilots because a flight must contain a pilot and a copilot.
- loadPeopleFromDisk and savePeopleToDisk: If a crew checking operation is in progress, those methods must wait for the end of that checking operation.

Question 1: Write the source code of enum CrewType.

```
public enum CrewType { PILOT, COPILOT, HOSTESS; }
```

Question 2: Write the source code of the class CrewMember.

```
import java.util.*;
public class CrewMember extends Person {
   private static final long serialVersionUID = 1L;
   private final CrewType type;
   private Date recruited;

   public CrewMember(String name, int ID, CrewType type) {
      super(name, ID);
      this.type = type; recruited = new Date( );
   }
   public void setRecruited(Date rec) { recruited = rec; }
   public CrewType getType() { return type; }
   public Date getRecruited() { return recruited; }
   public boolean isSenior( ) {
      Calendar rec =  Calendar.getInstance();
      rec.setTime(getRecruited());
      rec.add(Calendar.YEAR, FlightRules.yearsToBecomeSenior);
      Calendar now =  Calendar.getInstance();
      now.setTime(new Date());
      if( now.after(rec) ) return true;
      return false;
   }
}
```

Question 3: Write the source code of the class Passenger.

```
public class Passenger extends Person {
   private static final long serialVersionUID = 1L;
   private double luggageWeight;
   private Infant infant;
   public Passenger(String name, int ID) { super(name, ID); }
   public double getLuggageWeight() { return luggageWeight; }
   public void setLuggageWeight(double luggageWeight) {
            this.luggageWeight = luggageWeight; }
   public void setInfant( int ageInMonths ) { infant = new Infant( ageInMonths ); }
   public int getInfantMonths( ) {
      if( infant != null )  return infant.getAgeInMonths();
      else            return -1;
   }
   private class Infant {
      private int ageInMonths;
      public Infant(int ageInMonths) { this.ageInMonths = ageInMonths; }
      public int getAgeInMonths() { return ageInMonths; }
   }
}
```

Question 4: Write the source code of the exception FlightException.

```
public class FlightException extends RuntimeException {
   private static final long serialVersionUID = 1L;
   public FlightException(String arg0) {
      super(arg0);
   }
}
```

Question 5: Write the source code of the multithreaded class of CheckPilots.

```
public class CheckPilots implements Runnable {
   private ArrayList<Person> people;
   public CheckPilots( ArrayList<Person> people ) { this.people = people; }
   public void run() {
      boolean hasPilot = false, hasCoPilot = false;
      for( Person person : people ) {
         if( person instanceof CrewMember ) {
            CrewMember member = (CrewMember) person;
            if( member.getType() == CrewType.COPILOT ) hasCoPilot = true;
            if( member.getType() == CrewType.PILOT )hasPilot = true;
         }
      }
      if( !hasPilot )    throw new FlightException("No pilot assigned!");
      if( !hasCoPilot )  throw new FlightException("No copilot assigned!");
   }}
```

Question 6: Write the source code of the class Flight.

```
import java.util.*;
import java.io.*;
import java.util.concurrent.*;
public class Flight {
   private ArrayList<Person> people;
   private ExecutorService executor;

   public Flight() {
      people = new ArrayList<Person>();
      executor = Executors.newCachedThreadPool( );
   }
   public Person searchPerson( int ID ) {
      for( Person person : people )
         if( person.getID() == ID ) return person;
      return null;
   }
   public void addPerson( Person person ) {
      if( searchPerson(person.getID()) == null ) people.add(person);
      else throw new FlightException("Duplicate ID: " + person.getID());
   }
   public void checkCrew( ) {
      executor.execute( new CheckPilots(people) );
      executor.execute( new CheckHostesses(people) );
      executor.shutdown();
   }
   @SuppressWarnings("unchecked") //notlandırılmayacak
   public void loadPeopleFromDisk( String fileName ) {
      while(!executor.isTerminated());
      try {
         ObjectInputStream stream = new ObjectInputStream(new FileInputStream(fileName));
         people = (ArrayList<Person>)stream.readObject();
         stream.close();
      }
      catch (IOException e | ClassNotFoundException e) { e.printStackTrace(); }
   }
   public void savePeopleToDisk( String fileName ) {
      while(!executor.isTerminated());
      try {
         ObjectOutputStream stream = new ObjectOutputStream(new FileOutputStream(fileName));
         stream.writeObject(people);
         stream.close();
      }
      catch (IOException e) { e.printStackTrace(); }
   }
}
```