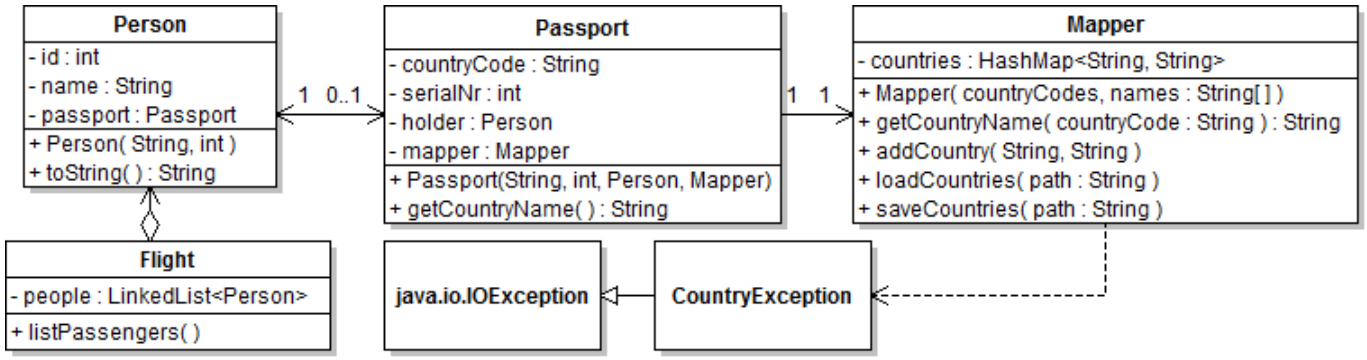
**BLM2562 OBJECT ORIENTED PROG., Section 1&2, 2nd MIDTERM (25/05/2017)**

| **Duration:** | 90mins. | | **Score:** | | | | **Student Nr:** | **Signature:** |
|---|---|---|---|---|---|---|---|---|
| **Grading:** | **1** | **2** | **3** | **4** | **5** | **6** | **Name, Surname:** | |
| | 10 | 35 | 10 | 10 | 25 | 10 | | |

## QUESTIONS




Answer these questions according to the UML class schema 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 class CountryException.

**Question 2:** Write the source code of class Mapper. This is used for relating country codes with country names, such as relating "TR" with "Türkiye". Explanation of its methods are as follows:

- The parameters supplied to the constructor will populate the HashMap.
- The getCountryName method will throw a CountryException if the given countryCode does not exist in the HashMap.
- The addCountry method will insert a new mapping into the HashMap. However, it will throw a CountryException if there is already a country with the given countryCode in the HashMap
- The loadCountries method loads the HashMap from a file.
- The saveCountries method saves the HashMap to a file.

**Question 3:** Write the source code of the getCountryName method of class Passport.

**Question 4:** Write the source code of the toString method of class Person. It should return the person's name and the name of the country of his/her passport.

**Question 5:** Write the source code of a class with a main method. It should create 3 person, 2 passports, one mapper and introduce each person **in a separate thread**. You may need to code additional regular classes or inner classes in your answer.

**Question 6:** Write the source code of the listPassengers method of class Flight.

**Question 1:** Write the source code of class UnknownDataException. (10p)

```
public class CountryException extends java.io.IOException {
   public CountryException( String e ) {
      super(e);
   }
}
```

**Question 2:** Write the source code of class Mapper. (35p)

```
import java.util.*;
import java.io.*;
public class Mapper {
   private HashMap<String, String> countries;

   public Mapper( String[] countryCodes, String[] names ) throws CountryException {
      countries = new HashMap<String, String>();
      for( int i = 0; i < names.length; i++ ) {
         addCountry(countryCodes[i], names[i]);
      }
   }
   public String getCountryName( String countryCode ) throws CountryException {
      String result = countries.get(countryCode);
      if( result == null )
         throw new CountryException("Non-existant country: " + countryCode);
      return result;
   }
   public void addCountry( String code, String name ) throws CountryException {
      if( countries.put(code, name) != null )
         throw new CountryException("Code already exists: " + code + ":" + name );
      countries.put(code, name);
   }
   @SuppressWarnings("unchecked")
   public void loadCountries( String path ) {
      try {
         ObjectInputStream input = new ObjectInputStream( new FileInputStream(path) );
         countries = (HashMap<String, String>) input.readObject();
         input.close();
      }
      catch (IOException e) {
         e.printStackTrace();
      }
      catch (ClassNotFoundException e) {
         e.printStackTrace();
      }
   }
   public void saveCountries( String path ) {
      try {
         ObjectOutputStream output = new ObjectOutputStream( new FileOutputStream(path) );
         output.writeObject(countries);
         output.close();
      }
      catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

**Question 3:** Write the source code of the getCountryName method of class Passport. (10p)

```
   public String getCountryName() throws CountryException {
      return mapper.getCountryName(countryCode);
   }
```

**Question 4:** Write the source code of the toString method of class Person. (10p)

```
public String toString( ) {
   String result = "Name: " + name;
   try {
      if( passport != null )
         result += "Nationality: " + passport.getCountryName();
   }
   catch (CountyException e) {
      e.printStackTrace();
   }
   return result;
}
```

**Question 5:** Write the source code of a class with a main method. ... (25p)

```
public class MainProgram {
   public static void main(String[] args) {
      Person yasar = new Person("Yaşar Nuri Öztürk",1111);
      Person nur = new Person("Cemalnur Sargut",2222);
      Person muriel = new Person("Muriel Maufroy",3333);
      String codes [ ] = { "TR", "FR", "RU", "CH" };
      String names [ ] = { "Türkiye", "Fransa", "Rusya", "Çin" };
      try {
         Mapper mapper = new Mapper( codes, names );
         Passport p1 = new Passport("TR", 101010, yasar, mapper);
         Passport p2 = new Passport("FR", 202020, muriel, mapper);
         yasar.setPassport(p1);
         muriel.setPassport(p2);
      }
      catch (CountryException e) {
         e.printStackTrace();
      }
      Thread t;
      t = new Thread( new IntroductionTask(yasar) );
      t.start();
      t = new Thread( new IntroductionTask(nur) );
      t.start ();
      t = new Thread( new IntroductionTask(muriel) );
      t.start ();
   }
}

//This class can be an inner class of MainProgram, too.
class IntroductionTask implements Runnable {
   private final Person aPerson;

   public IntroductionTask(Person aPerson) {
      this.aPerson = aPerson;
   }
   public void run() {
      System.out.println( aPerson );
   }
}
```

**Question 6:** Write the source code of the listPassengers method of class Flight. (10p)

```
   public void listPassengers( ) {
      System.out.println("Passenger list of flight "+code);
      for( Person p : people )
         System.out.println( p );
   }
```

**Question 5:** Alternative solution by using anonymous inner runnable classes:

```
public class MainProgramV2 {
	public static void main(String[] args) {
		Person yasar = new Person("Yaşar Nuri Öztürk",1111);
		Person nur = new Person("Cemalnur Sargut",2222);
		Person muriel = new Person("Muriel Maufroy",3333);
		String codes [ ] = { "TR", "FR", "RU", "CH" };
		String names [ ] = { "Türkiye", "Fransa", "Rusya", "Çin" };
		try {
			Mapper mapper = new Mapper( codes, names );
			Passport p1 = new Passport("TR", 101010, yasar, mapper);
			Passport p2 = new Passport("FR", 202020, muriel, mapper);
			yasar.setPassport(p1);
			muriel.setPassport(p2);
		}
		catch (CountryException e) {
			e.printStackTrace();
		}

		Thread t;
		t = new Thread( new Runnable() {
			@Override
			public void run() {
				System.out.println(yasar);
			}
		} );
		t.start();
		t = new Thread( new Runnable() {
			@Override
			public void run() {
				System.out.println(nur);
			}
		} );
		t.start ();
		t = new Thread( new Runnable() {
			@Override
			public void run() {
				System.out.println(muriel);
			}
		} );
		t.start ();
	}
}
//görüldüğü üzere fazla uzatıyor, kişileri dizi yaparak biraz daha kısaltılabilir.
```

**Question 5:** Alternative solution by extending Person and implementing Runnable:

This will be tricky: A person has a passport, too. Just sending name and ID of a person is not enough. Passport information must be sent, too. Alternatively, a complete person instance can be sent, too. Examine PersonRunnerBuggy and PersonRunnerCorrect carefully.

```
public class MainProgramV3 {
	public static void main(String[] args) {
		Person yasar = new Person("Yaşar Nuri Öztürk",1111);
		Person nur = new Person("Cemalnur Sargut",2222);
		Person muriel = new Person("Muriel Maufroy",3333);
		String codes [ ] = { "TR", "FR", "RU", "CH" };
		String names [ ] = { "Türkiye", "Fransa", "Rusya", "Çin" };
		try {
			Mapper mapper = new Mapper( codes, names );
			Passport p1 = new Passport("TR", 101010, yasar, mapper);
			Passport p2 = new Passport("FR", 202020, muriel, mapper);
			yasar.setPassport(p1);
			muriel.setPassport(p2);
		}
```

```
            catch (CountryException e) {
               e.printStackTrace();
            }
            Thread t;
            t = new Thread( new PersonRunnerCorrect(yasar) );
            t.start();
            t = new Thread( new PersonRunnerBuggy(nur.getName(), nur.getId()) );
            t.start ();
            t = new Thread( new PersonRunnerBuggy(muriel.getName(), muriel.getId()) );
            t.start ();

      }
}

class PersonRunnerBuggy extends Person implements Runnable {
      public PersonRunnerBuggy(String name, int id) {
         super(name, id);
      }
      public void run( ) {
         System.out.println(this);
      }
}

class PersonRunnerCorrect extends Person implements Runnable {
      public PersonRunnerCorrect(Person aPerson) {
         super(aPerson.getName(), aPerson.getId());
         setPassport(aPerson.getPassport());
      }
      public void run( ) {
         System.out.println(this);
      }
}
```