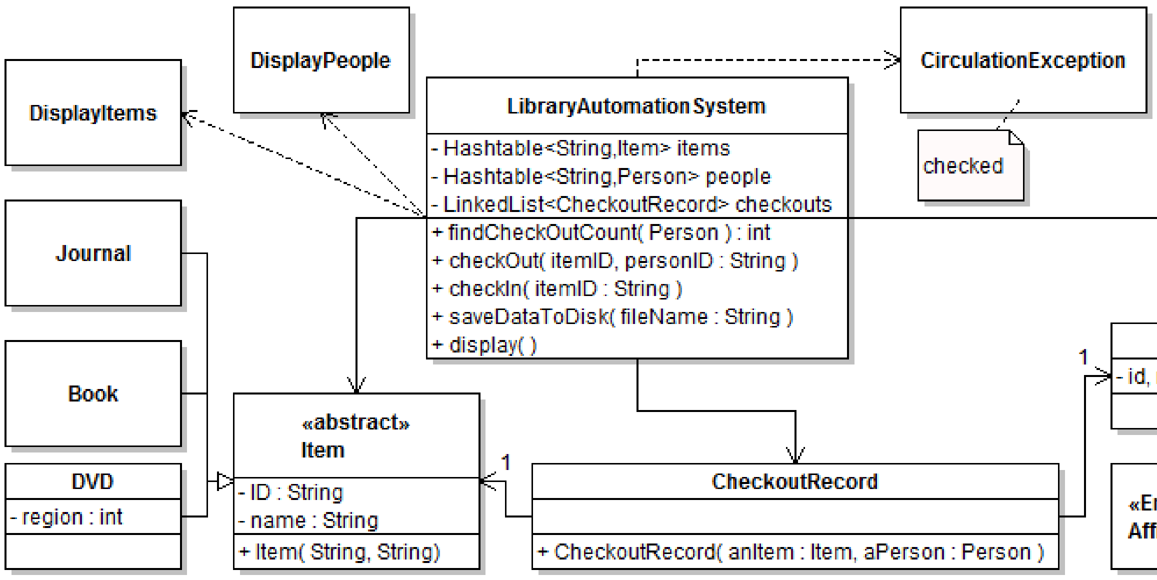
BLM2562 OBJECT ORIENTED PROG., Section 2&3, FINAL EXAM **(30/05/2015)**

Duration:	90 mins.			**Score:**				**Student Nr:**	**Signature:**
Grading:	**1**	**2**	**3**	**4**	**5**	**6**	**Group**	**Name, Surname:**	
	10	15	10	20	45				

QUESTIONS


DisplayItems
DisplayPeople
LibraryAutomationSystem
- Hashtable<String,Item> items
- Hashtable<String,Person> people
- LinkedList<CheckoutRecord> checkouts
+ findCheckOutCount(Person) : int
+ checkOut(itemID, personID : String)
+ checkIn(itemID : String)
+ saveDataToDisk(fileName : String)
+ display( )
CirculationException
checked
Journal
Book
DVD
- region : int
«abstract»
Item
- ID : String
- name : String
+ Item(String, String)
1
CheckoutRecord
+ CheckoutRecord(anItem : Item, aPerson : Person)
1
Person
- id, name : String
«Enum»
AffiliationType

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. You don’t have to consider inconsistencies other than the ones explicitly stated in questions. You will be writing some part of a library information system.

Question 1: Write the source code of class CirculationException.

Question 2: Write the source code of classes AffiliationType and Person. A person can be affilliated with a library as being a student, a teacher or a staff member. Those person types can borrow maximum 3, 6 and 4 items during any time period, respectively.

Question 3: Write the source code of classes Item and DVD.

Question 4: Write the source code of class DisplayPeople, which is responsible from displaying the elements in the people table. The DisplayPeople instance must list students first, then list the staff and finally the teachers, separately. Class LibraryAutomationSystem will create one instance of this class and execute it in multithreaded fashion in its display method. You will code LibraryAutomationSystem.display in the next question.

Question 5: Write the source code of class LibraryAutomationSystem. Explanation of its methods is given below.

- findCheckOutCount: Find how many items that this person with given ID currently has (checked out).
- checkOut: This method lends an item to a person. However, a detailed CirculationException must be generated if one of the following is attempted:
    - A person or an Item with given ID does not exist.
    - An item which is already lent (checked out).
    - The person has reached his/her checkout limit.
- checkIn: This method deletes the related check out record from the list. However, a detailed CirculationException must be generated if one of the following is attempted:
    - An Item with given ID does not exist.
    - The item has not been checked out before.
- saveDataToDisk: Saves the state information (people, items and check out records) to disk. However, it must wait for the display operation to be finished before saving.

Question 1: Write the source code of class CirculationException.

```
import java.io.IOException;
public class CirculationException extends IOException {
  public CirculationException() { super(); }
  public CirculationException(String arg0) { super(arg0); }
}
```

Question 2: Write the source code of classes AffiliationType and Person.

```
public enum AffiliationType {
  STUDENT(3), TEACHER(6), STAFF(4);
  private int itemLimit;
  private AffiliationType( int itemLimit ) {
    this.itemLimit = itemLimit;
  }
  public int getItemLimit() { return itemLimit; }
}
public class Person implements java.io.Serializable {
  private static final long serialVersionUID = 1L;
  private String id, name;
  private AffiliationType type;
  public Person(String id, String name, AffiliationType type) {
    this.id = id;    this.name = name;    this.type = type;
  }
  public String getId() { return id; }
  public String getName() { return name; }
  public AffiliationType getType() { return type; }
}
```

Question 3: Write the source code of classes Item and DVD.

```
public abstract class Item implements java.io.Serializable {
  private static final long serialVersionUID = 1L;
  private String id, name;
  public Item(String id, String name) {
    this.id = id; this.name = name;
  }
  public String getId() { return id; }
  public String getName() { return name; }
}
public class DVD extends Item {
  private static final long serialVersionUID = 1L;
  private int regionCode;
  public DVD(String id, String name, int regionCode) {
    super(id, name);
    this.regionCode = regionCode;
  }
  public int getRegionCode () { return regionCode; }
}
```

Question 4: Write the source code of class DisplayPeople

```java
import java.util.Hashtable;
public class DisplayPeople implements Runnable {
  private Hashtable<String, Person> people;

  public DisplayPeople(Hashtable<String, Person> people) {
    this.people = people;
  }
  public void run() {
    System.out.println("STAFF LIST: ");
    System.out.println("--------------------------------");
    for( Person aPerson : people.values() ) {
      if( aPerson.getType() == AffiliationType.STAFF)
      System.out.println( aPerson.getId() + "\t"
          + aPerson.getName() + "\t" );
    }
    System.out.println("STUDENT LIST: ");
    System.out.println("--------------------------------");
    for( Person aPerson : people.values() ) {
      if( aPerson.getType() == AffiliationType.STUDENT)
      System.out.println( aPerson.getId() + "\t"
          + aPerson.getName() + "\t" );
    }
    System.out.println("TEACHER LIST: ");
    System.out.println("--------------------------------");
    for( Person aPerson : people.values() ) {
      if( aPerson.getType() == AffiliationType.TEACHER)
      System.out.println( aPerson.getId() + "\t"
          + aPerson.getName() + "\t" );
    }

  }
}
```

Question 5: Write the source code of class LibraryAutomationSystem. Explanation of its methods is given below.

```java
import java.util.*;
import java.io.*;
import java.util.concurrent.*;
public class LibraryAutomationSystemV3 {
  private Hashtable<String, Item> items;
  private Hashtable<String, Person> people;
  private LinkedList<CheckoutRecord> checkouts;
  private ExecutorService pool;

  public LibraryAutomationSystemV3( ) {
    items = new Hashtable<String, Item>( );
    people = new Hashtable<String, Person>( );
    checkouts = new LinkedList<CheckoutRecord>( );
    pool = Executors.newCachedThreadPool( );
    //veya load/save/display tümü synchronized
  }
  public int findCheckOutCount( Person aPerson ) {
    int result = 0;
    for( CheckoutRecord aRecord : checkouts )
      if( aRecord.getPerson() == aPerson )
        result++;
    return result;
  }
  public void checkOut(String itemID, String personID) throws CirculationException {
    Person aPerson = people.get(personID);
    if( aPerson == null )
      throw new CirculationException("Unknown personID "+personID);
    Item anItem = items.get(itemID);
    if( anItem == null )
      throw new CirculationException("Unknown itemID "+itemID);
    for( CheckoutRecord aRecord : checkouts )
      if( aRecord.getItem() == anItem )
        throw new CirculationException("Item already out: "+itemID);
    if( findCheckOutCount(aPerson) == aPerson.getType().getItemLimit() )
      throw new CirculationException("Person " + personID +
          " has reached to the checkout limit");
    CheckoutRecord newRecord = new CheckoutRecord(aPerson, anItem, new Date() );
    checkouts.add(newRecord);
  }
  public void checkIn( String itemID ) throws CirculationException {
    Item anItem = items.get(itemID);
    if( anItem == null )
      throw new CirculationException("Unknown itemID "+itemID);
    for( CheckoutRecord aRecord : checkouts ) {
      if( aRecord.getItem() == anItem ) {
        checkouts.remove(aRecord); return;
      }
    }
    throw new CirculationException("Item not checked out: "+itemID);
  }
  public void saveDataToDisk( String fileName ) {
    while( pool.isTerminated() );
    //veya load/save/display tümü synchronized
    try {
      ObjectOutputStream str = new ObjectOutputStream(
          new FileOutputStream(fileName));
      str.writeObject(items);
      str.writeObject(people);
      str.writeObject(checkouts);
      str.close();
    }
    catch(IOException e) { e.printStackTrace(); }
  }
  public void display( ) {
    pool.execute( new DisplayPeople(people) );
  }
  public boolean addPerson( Person aPerson ) {
    if( people.get(aPerson.getId()) == null ) {
      people.put(aPerson.getId(), aPerson);
      return true;
    }
    return false;
  }
  public boolean addItem( Item anItem ) {
    if( items.get(anItem.getId()) == null ) {
      items.put(anItem.getId(), anItem);
      return true;
    }
    return false;
  }
}
```