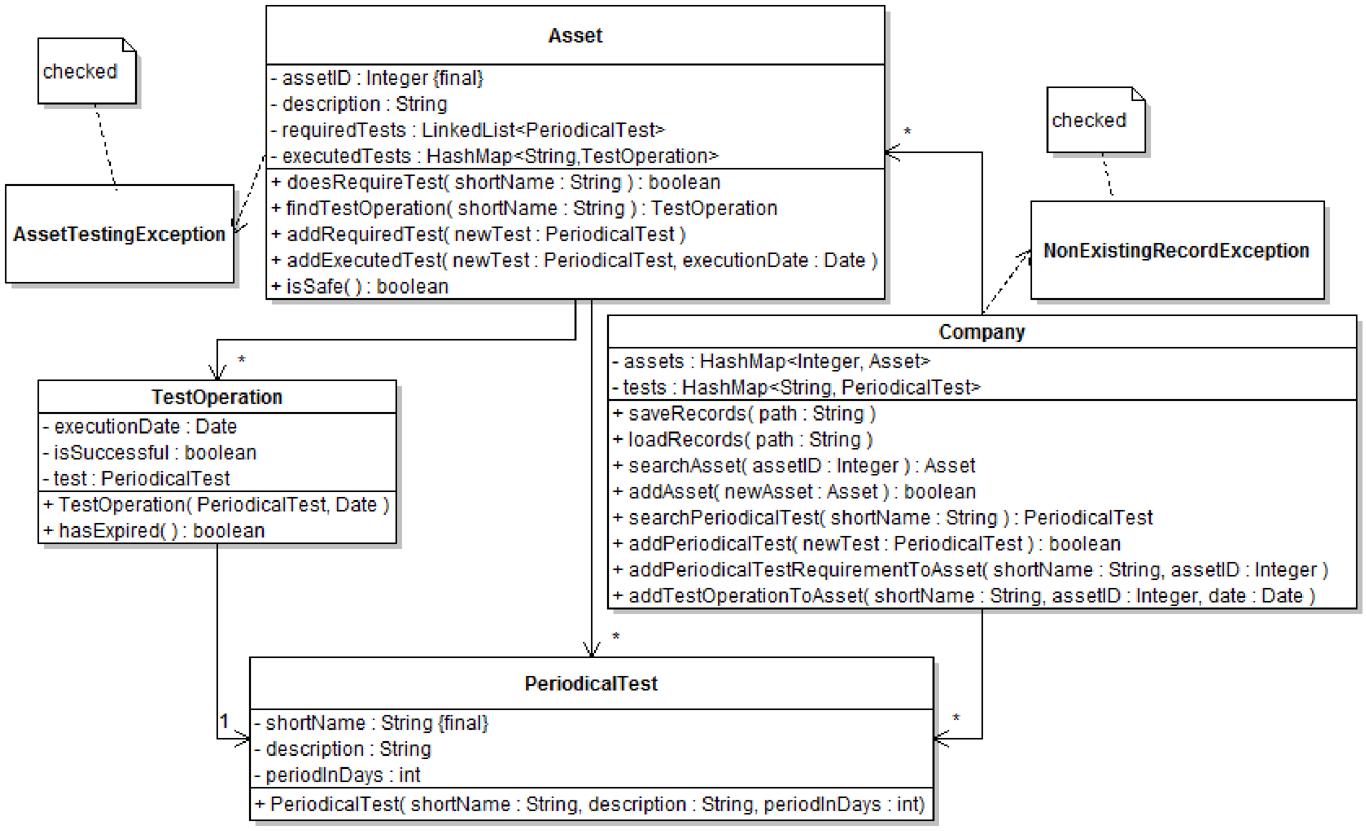
BLM2562 OBJECT ORIENTED PROG., Gr.1(YES)&Gr.2(MSA), 1ST MIDTERM (13/04/2017)

Duration:	90 mins.		**Score:**					**Student Nr:**	**Signature:**
Grading:	**1**	**2**	**3**	**4**	**5**	**6**	**Group**	**Name, Surname:**	
	10	20	40	30					

QUESTIONS


checked
AssetTestingException
Asset
- assetID : Integer {final}
- description : String
- requiredTests : LinkedList<PeriodicalTest>
- executedTests : HashMap<String,TestOperation>
+ doesRequireTest(shortName : String) : boolean
+ findTestOperation(shortName : String) : TestOperation
+ addRequiredTest(newTest : PeriodicalTest)
+ addExecutedTest(newTest : PeriodicalTest, executionDate : Date)
+ isSafe( ) : boolean
*
checked
NonExistingRecordException
*
TestOperation
- executionDate : Date
- isSuccessful : boolean
- test : PeriodicalTest
+ TestOperation(PeriodicalTest, Date)
+ hasExpired( ) : boolean
Company
- assets : HashMap<Integer, Asset>
- tests : HashMap<String, PeriodicalTest>
+ saveRecords(path : String)
+ loadRecords(path : String)
+ searchAsset(assetID : Integer) : Asset
+ addAsset(newAsset : Asset) : boolean
+ searchPeriodicalTest(shortName : String) : PeriodicalTest
+ addPeriodicalTest(newTest : PeriodicalTest) : boolean
+ addPeriodicalTestRequirementToAsset(shortName : String, assetID : Integer)
+ addTestOperationToAsset(shortName : String, assetID : Integer, date : Date)
*
PeriodicalTest
1
- shortName : String {final}
- description : String
- periodInDays : int
+ PeriodicalTest(shortName : String, description : String, periodInDays : int)
*

Answer the following 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.

Domain information: A company has multiple assets distinguished from each other by their assetID fields. Each asset require some possibly overlapping periodical tests distinguished from each other by their shortName fields. An asset, such as a fire extinguisher having assetID=1 should be tested for its pressure (Short name=”PrT”, description=”Pressure Test”) and it must have an execution test (Short name=”ExT”, description=”Execution test”). Another asset, a fire alarm having assetID=2 must only have an execution test. Whenever a test is actually done on a particular asset, a relevant TestOperation object is created and associated with the asset. A test operation is expired if its periodical test was done more days before than the periodical test's period. An asset is safe if all its required tests are done and not expired.

Question 1: Write the source code of class AssetTestingException.

Question 2: Write the source code of class TestOperation.

Question 3: Write the source code of class Asset. The methods addRequiredTest and addExecutedTest can generate an exception with detailed information

Question 4: The details of some methods of the class Company are given below. Write the source codes of only those methods.

- saveRecords: Saves the generic HashMaps assets and tests to the path given as its parameter.
- addPeriodicalTestRequirementToAsset: Carries out the task. Generates an exception with detailed information if necessary.

Question 1: Write the source code of class AssetTestingException.

```
public class AssetTestingException extends java.io.IOException {
   public AssetTestingException(String message) {
      super(message);
   }
}
```

Question 2: Write the source code of class TestOperation.

```
public class TestOperation implements java.io.Serializable {
   private static final long serialVersionUID = 1L;
   private PeriodicalTest test;
   private Date executionDate;

   public TestOperation(PeriodicalTest test, Date executionDate) {
      this.test = test;  this.executionDate = executionDate;
   }
   public PeriodicalTest getTest() { return test; }
   public Date getExecutionDate() { return executionDate; }
   public boolean hasExpired( ) {
      Calendar cal = Calendar.getInstance();
      cal.setTime(executionDate);
      cal.add(Calendar.DAY_OF_MONTH, test.getPeriodInDays());
      Date expireDate = cal.getTime();
      Date today = new Date();
      return expireDate.before(today);
   }
}
```

Question 4: The details of some methods of the class Company are given below.

```
   public void saveRecords( String path ) {
      try {
         ObjectOutputStream str = new ObjectOutputStream(
               new FileOutputStream(path) );
         str.writeObject(assets);
         str.writeObject(tests);
         str.close();
      }
      catch (IOException e) { e.printStackTrace(); }
   }
   public void addPeriodicalTestRequirementToAsset( String shortName, Integer assetID )
         throws NonExistingRecordException {
      Asset asset = assets.get(assetID);
      if( asset == null )
         throw new NonExistingRecordException("Asset with ID " + assetID + " does not exist");
      PeriodicalTest test = tests.get(shortName);
      if( test == null )
         throw new NonExistingRecordException("Test " + shortName + " does not exist");
      try {
         asset.addRequiredTest(test);
      }
      catch (AssetTestingException e) { e.printStackTrace(); }
   }
   public void addPeriodicalTestRequirementToAssetV2( String shortName, Integer assetID ) {
      try {
         Asset asset = searchAsset(assetID);
         if( asset == null )
            throw new NonExistingRecordException("Asset with ID " + assetID + " does not exist");
         PeriodicalTest test = searchPeriodicalTest(shortName);
         if( test == null )
            throw new NonExistingRecordException("Test " + shortName + " does not exist"); //1:
         asset.addRequiredTest(test); //burası 2:AssetTestingException atabileceğinden
      }
      catch (NonExistingRecordException | AssetTestingException e) {//Exception e de 1:&2: kapsar
         e.printStackTrace();
      }
   }
```

Question 3: Write the source code of class Asset.

```
import java.util.*;
public class Asset implements java.io.Serializable {
   private static final long serialVersionUID = 1L;
   private final Integer assetID; private String description;
   private LinkedList<PeriodicalTest> requiredTests;
   private HashMap<String, TestOperation> executedTests;

   public Asset(Integer assetID, String description) {
      this.assetID = assetID; this.description = description;
      requiredTests = new LinkedList<PeriodicalTest>();
      executedTests = new HashMap<String, TestOperation>();
   }
   public Integer getAssetID() { return assetID; }
   public String getDescription() { return description; }
   public void setDescription(String description) { this.description = description; }
   public boolean doesRequireTest( String shortName ) {
      for( PeriodicalTest test : requiredTests )
         if( test.getShortName().equalsIgnoreCase(shortName) )
            return true;
      return false;
   }
   public TestOperation findTestOperation( String shortName ) {
      return executedTests.get(shortName);
   }
   public void addRequiredTest( PeriodicalTest newTest ) throws AssetTestingException {
      if( doesRequireTest(newTest.getShortName()) )
         throw new AssetTestingException( "Asset with ID " + assetID
               + " already requires test " + newTest.getShortName() );
      requiredTests.add(newTest);
   }
   public void addExecutedTest(PeriodicalTest newTest,Date execDate) throws AssetTestingException{
      if( !doesRequireTest(newTest.getShortName()) )
         throw new AssetTestingException( "Asset with ID " + assetID
               + " does not require test " + newTest.getShortName() );
      TestOperation newOperation = new TestOperation(newTest, execDate);
      executedTests.put(newTest.getShortName(), newOperation);
   }
   public boolean isSafe( ) {
      for( PeriodicalTest required : requiredTests ) {
         TestOperation done = executedTests.get(required.getShortName());
         if( done == null || done.hasExpired() )
            return false;
      }
      return true;
   }
}
```