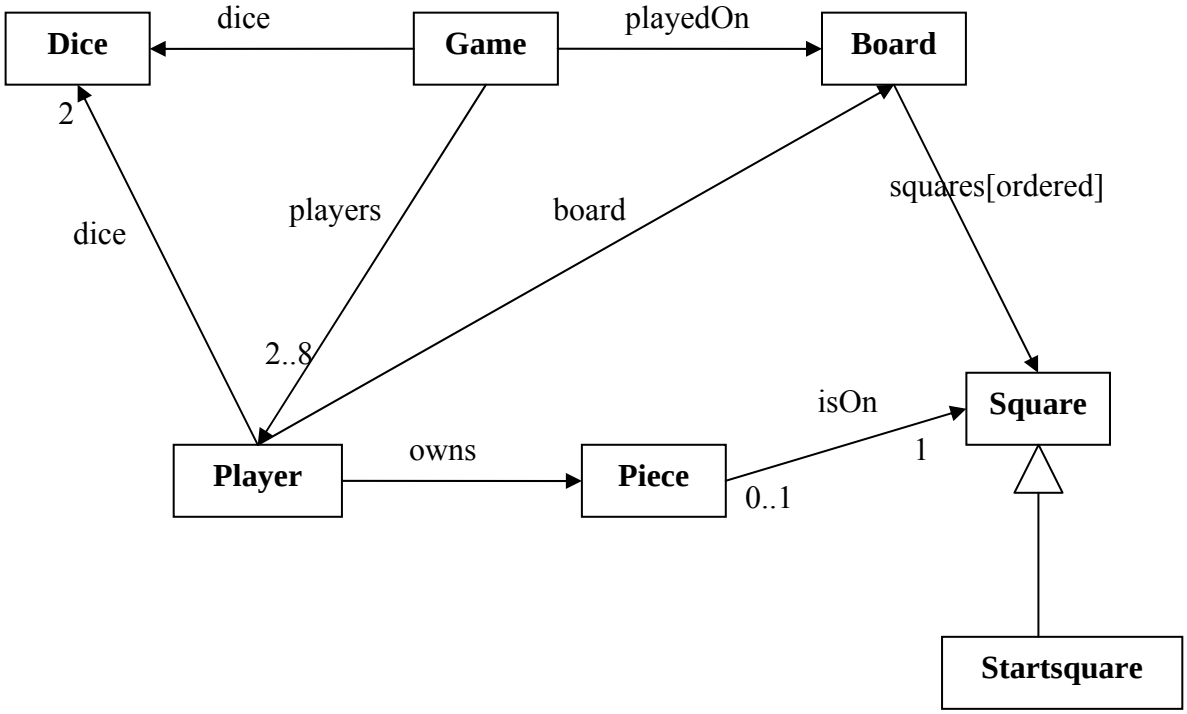
SOLUTION OF MONOPOLY QUESTION

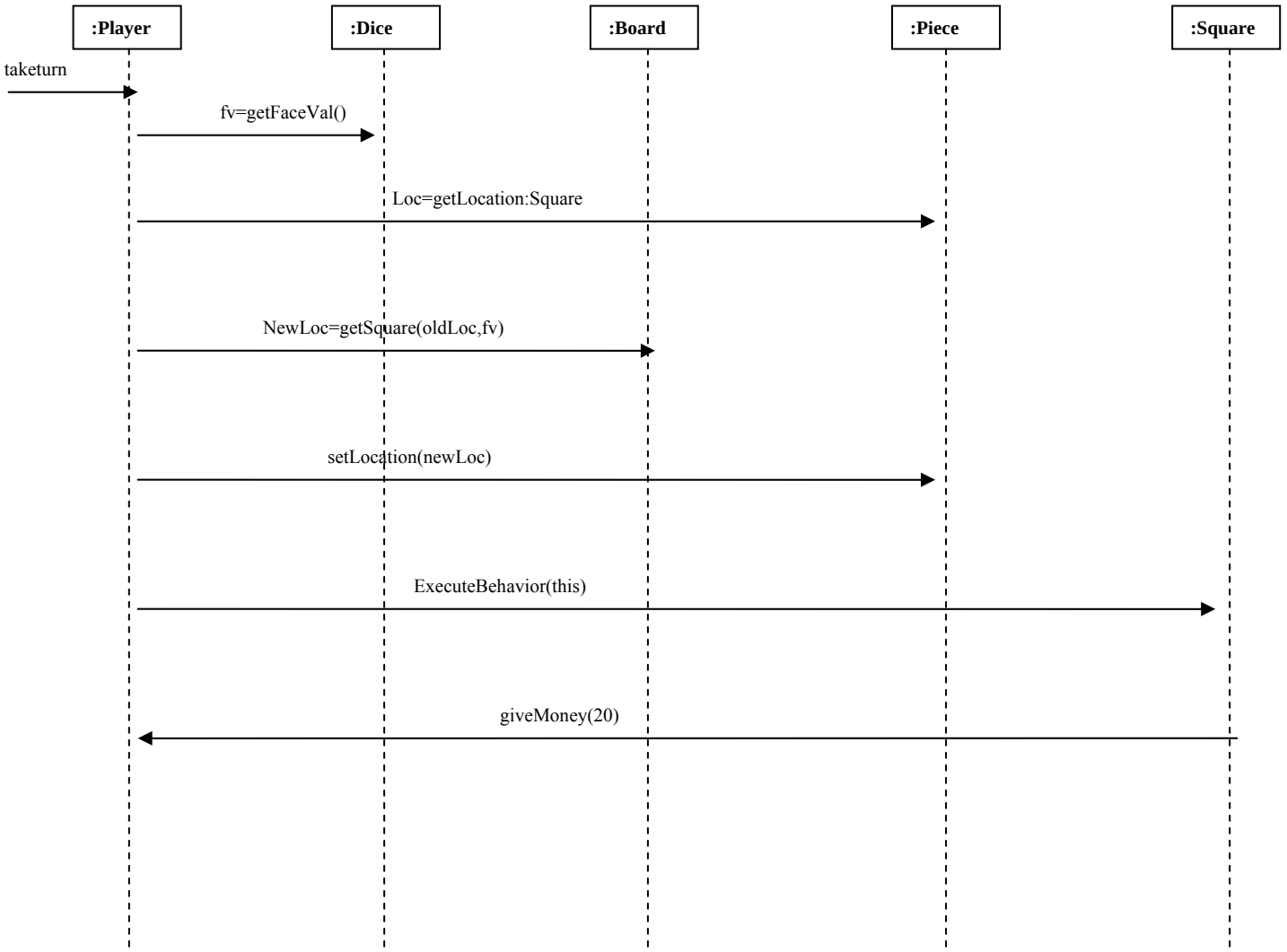
1) **Considering the Monopoly example in your textbook, assume that there is a special square called "Start". If a player's piece comes to this special square, she gets 20 TL bonus. For this situation**

i) Define the domain model (15 p),



Note: All relations must be labelled, direction should also be indicated

ii) Define the interaction diagram for the "takeTurn" event (25 p).



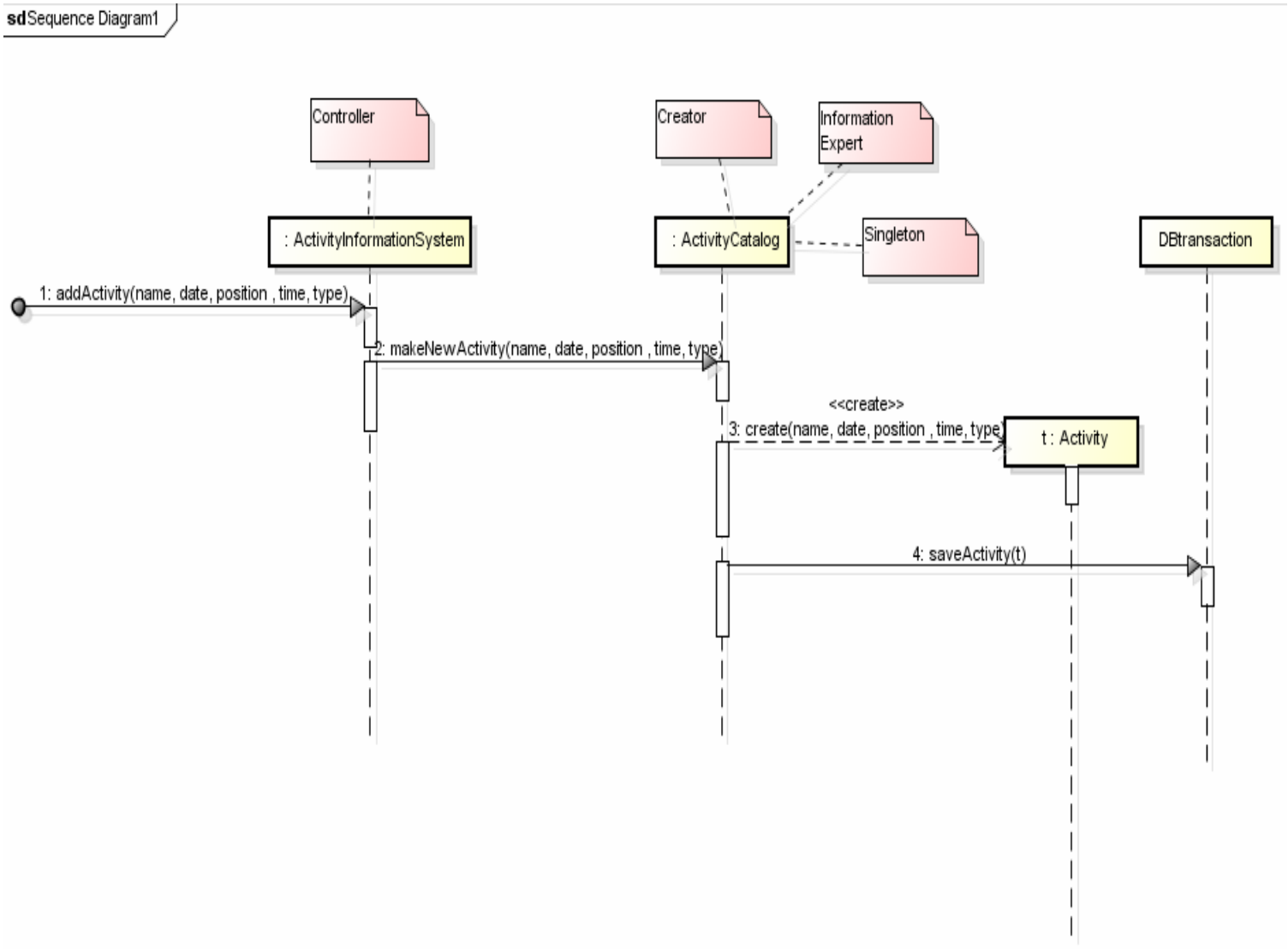
Key issues in design:

- Player controls piece
- Board knows square positions
- Square knows how to give or take Money, and similar behavior on player
- ExecuteBehavior is an abstract (virtual) method call (Polymorphism)

Player: Controller but expert also OK
Board: Info Expert
Square: Info Expert, Polymorphism
Dice: Info Expert

2-)

ii) Draw the interaction diagram for the "Add theatre performance" event and discuss in the context of GRASP patterns.



iii) Write unite tests for the classes you have designed.

Test1

```
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;
import java.util.Random;
import java.util.UUID;
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

import junit.framework.TestCase;


public class TiyatroTest extends TestCase {

	private List<Object> testObjects;
	private Helper helper=new Helper();

	protected void setUp() throws Exception {
		super.setUp();
		testObjects = new ArrayList();
		}

	private void deleteAllTestObjects() {

		Iterator i=testObjects.iterator();
		while (i.hasNext())
		{
			try {

				Deletable deletableObj=(Deletable)i.next();
				deletableObj.finalize();
			} catch (Throwable e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}

		}
	}


	public void testTiyatro_EmptyCreation()
```

```
	{
		Tiyatro tiyatro=helper.createAnonymousEmptyTiyatro();
		addTestObject(tiyatro);
		assertNameIsEqualToBos(tiyatro);
		tearDown();
	}

	public void testTiyatro_ParameterizedCreation()
	{
		String expectedName=helper.createAnonymousName();
		Mekan expectedMekan=helper.createAnonymousMekan();
		Date expectedDate=helper.createAnonymousDate();
		Tiyatro tiyatro=helper.createAnonymousParameterizedTiyatro(expectedName,
expectedDate,expectedMekan);
		addTestObject(tiyatro);
		assertNameEqualsExpectedName(tiyatro.getName(), expectedName);
		tearDown();
	}
	private void assertNameEqualsExpectedName(String tiyatro,
			String expectedName) {
		assertEquals(tiyatro, expectedName);

	}
	public void addTestObject(Object obj)
	{
		testObjects.add(obj);
	}
	private void assertNameIsEqualToBos(Tiyatro tiyatro)
	{
		assertEquals(tiyatro.getName(),"BOŞ");
	}
	public void tearDown()
	{
		deleteAllTestObjects();
	}

}
```

Test2

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;

import junit.framework.TestCase;



public class DBClassTest extends TestCase {

	private Helper helper=new Helper();

	private List<Object> testObjects;

	protected void setUp() throws Exception {
		super.setUp();
		testObjects = new ArrayList();
		}

	public void testTiyatroEkle()
	{
		Tiyatro tiyatro=helper.createAnonymousTiyatro();
		addTestObject(tiyatro);
		//Oğuz Hoca sınavda sizden tek bir veritabanı sınıfı istendiğini belirttiği için veritabanı aşağıda tek bir sınıfla temsil edilmektedir.
		DBClass db=new DBClass();
		addTestObject(db);
		db.addTiyatro(tiyatro);
		 try {
```

```
            String url = "jdbc:mysql://localhost:3306/EtkinlikDB";
            Connection conn = DriverManager.getConnection(url,"","");
            Statement stmt = conn.createStatement();
            ResultSet rs;

            rs = stmt.executeQuery("SELECT * FROM Tiyatrolar WHERE name="+tiyatro.getName());
            while ( rs.next() ) {
             String date = rs.getString("date");
             SimpleDateFormat sdf = new SimpleDateFormat("dd-MMM-yyyy HH:mm:ss", Locale.ROOT);
             assertDatesAreTheSame(tiyatro.getDate(),sdf.parse(date));
             String mekan=rs.getString("mekan");
             assertMekanlarAynı(tiyatro.getMekan().toString(),mekan);
            }
            conn.close();
        } catch (Exception e) {
            System.err.println("Exception! ");
            System.err.println(e.getMessage());
        }


}
private void assertMekanlarAynı(String string, String mekan) {
            assertEquals(string, mekan);

}


private void assertDatesAreTheSame(Date date, Date parse) {

            assertEquals(date, parse);
}


public void addTestObject(Object obj)
{
            testObjects.add(obj);
}
```

```
private void deleteAllTestObjects() {

		Iterator i=testObjects.iterator();
		while (i.hasNext())
		{

			try {

				Deletable deletableObj=(Deletable)i.next();
				deletableObj.finalize();
			} catch (Throwable e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}

		}
	}


	public void tearDown()
	{
		deleteAllTestObjects();
	}
}
```

Helper

```
import java.util.Calendar;
import java.util.Date;
import java.util.Random;
import java.util.UUID;


public class Helper {


	public Tiyatro createAnonymousTiyatro() {

		String expectedName=createAnonymousName();
		Mekan expectedMekan=createAnonymousMekan();
		Date expectedDate=createAnonymousDate();
		return new Tiyatro(expectedName, expectedDate, expectedMekan);
	}


	public Tiyatro createAnonymousParameterizedTiyatro(String expectedName,
			Date expectedDate, Mekan expectedMekan) {

		return new Tiyatro(expectedName, expectedDate, expectedMekan);
	}



		public Mekan createAnonymousMekan() {

			return new Mekan(UUID.randomUUID().toString());
		}

		public String createAnonymousName() {

			return  UUID.randomUUID().toString();
		}
```

```
        public Date createAnonymousDate() {


            Random random = new Random();
            Calendar calendar = Calendar.getInstance();
            calendar.add(Calendar.DATE, 365 - random.nextInt(730));
            return calendar.getTime();

        }

    public Tiyatro createAnonymousEmptyTiyatro()
    {
        return new Tiyatro();
    }


}
```