Topic: Single Class

Question 1:

a. Design a class which draws a rectangle made of asterixes to the console area. The class must contain at least one method which determines the side's lengths of the rectangle without using nextInt method. Draw the UML class schema of your design.

b. A rectangle can be theoretically in any sizes. However, a typical command console has at most 24 lines and 80 rows. Modify your design and write its Java source code accordingly. You don't have to redraw the UML class schema.

c. Add a main method to your class which lets a user to draw rectangles. Use nextInt methods to obtain the side lengths from the user.

d. Draw the UML sequence diagram of the main method.

Question 2:

You need to write a utility class in order to determine the validity of different information. Code your class accordingly with the following functionality:

a. The last digit of ISBN numbers given to books is used as a check digit. Let x_1, x_2, ... x_{10} be the digits of an ISBN-10 number from left to right. In order to be valid, the digits must satisfy the following equation:

- $(10 * x_1 + 9 * x_2 + 8 * x_3 + 7 * x_4 + 6 * x_5 + 5 * x_6 + 4 * x_7 + 3 * x_8 + 2 * x_9 + x_{10}) \bmod 11 = 0$.
- If the last digit is given as X in the ISBN, then 10 is used as the value of x_{10}.

b. A valid e-mail address must contain only one @ and at least one dot.

Hints:

- You can check the javadoc for the following methods of the String class:
  i. public char[]toCharArray()
  ii. public static String valueOf(char c)
  iii. public int indexOf(String ch)
  iv. public int indexOf(String ch, int fromIndex)
  v. public int lastIndexOf(String ch)
- You can check the javadoc for the following methods of the Integer class:
  i. public static Integer valueOf(String s)

Answers to Question 1 (partial, different solutions exist as well):

```
package examples01;
import java.util.Scanner;
public class Rectangle {
   private int sideA, sideB;
   private final static int maxSideA = 24, maxSideB = 80;
   public Rectangle(int sideA, int sideB) {
      if( sideA > maxSideA )
         this.sideA = maxSideA;
      else
         this.sideA = sideA;
      if( sideB > maxSideB )
         this.sideB = maxSideB;
      else
         this.sideB = sideB;
   }
   public void setSideA( int sideA ) {
      if( sideA > maxSideA )
         this.sideA = maxSideA;
      else
         this.sideA = sideA;
   }
   public void setSideB( int sideB ) {
      if( sideB > maxSideB )
         this.sideB = maxSideB;
      else
         this.sideB = sideB;
   }
   public void draw() {
      for( int i=0; i<sideA; i++ ) {
         for( int j=0; j<sideB; j++ ) {
            if( i == 0 || i == sideA-1
                  || j == 0 || j == sideB-1)
               System.out.print("*");
            else
               System.out.print(" ");
         }
         System.out.println();
      }
   }
   public static void main( String[] args ) {
      Scanner in = new Scanner( System.in );
      System.out.print("Enter side A: ");
      int a = in.nextInt();
      System.out.print("Enter side B: ");
      int b = in.nextInt();
      Rectangle r = new Rectangle( a, b );
      r.draw();
      in.close();
   }
}
```

Answers to Question 1 (an alternative):

```java
package examples01;
import java.util.*;
public class Rectangle {
   private int rows, cols;
   public final static int maxRow = 24, maxCol = 80;

   public Rectangle( int rows, int cols ) {
      this.rows = rows; this.cols = cols;
      if( !isRowValid() )
         this.rows = maxRow;
      if( !isColValid() )
         this.cols = maxCol;
   }
   public int getRows() { return rows; }
   public int getCols() { return cols; }
   public void draw() {
      for( int i=0; i<rows; i++ ) {
         for( int j=0; j<cols; j++ ) {
            System.out.print("*");
         }
         System.out.println();
      }
   }
   public boolean isRowValid() {
      if( rows > 0 && rows <= maxRow )
         return true;
      return false;
   }
   public boolean isColValid() {
      if( cols > 0 && cols <= maxCol )
         return true;
      return false;
   }
   public static void main( String[] args ) {
      Rectangle r1 = new Rectangle(30, 88);
      r1.draw();
      if( r1.isColValid() && r1.isRowValid() )
         System.out.println("Test1 OK");
      else
         System.out.println("Test1 Fail");
      System.out.println("Rows: " + r1.getRows()
            + ", Cols: " + r1.getCols() );
   }
}
```

Topic: 1-1 Associated Classes

Question 1: A missile has a range, measured in kilometers, and weight, measured in kilograms. A particular missile is built for targets on the ground or targets in the air. Once built, those three properties of a missile cannot be altered. Write the Java source code of the missile class and draw its UML schema.

Question 2: Each aircraft has an empty weight and a maximum take-off weight, each measured in kilograms. If an aircraft is loaded over its capacity, it cannot fly. An aircraft must be flying in order to fire a missile. An unmanned aerial vehicle (UAV) is an aircraft which does not carry a human pilot. Some UAVs some cannot carry missiles but some can, albeit they can carry missiles of only one type and up to a limited number. Write the Java source code of the UAV class by using all the given information.

Question 3: Write the source code of a Java class with a main method in order to test the classes you have coded in previous questions. Try to take at least one numerical input from the user in the main method. Creating one instance for each class and firing a missile suffice.

Question 4: Draw a detailed UML class diagram of all classes you have coded so far.

Question 5: Draw the UML sequence diagram of the method you have coded for firing a missile.

Answers (partial, different solutions exist as well):

Question 1:

```java
public class Missile {
    private final int range, weight;
    private final boolean forAirTargets;

    public Missile(int range, int weight, boolean forAirTargets) {
        this.range = range;
        this.weight = weight;
        this.forAirTargets = forAirTargets;
    }
    public int getRange() { return range; }
    public int getWeight() { return weight; }
    public boolean isForAirTargets() { return forAirTargets; }
}
```

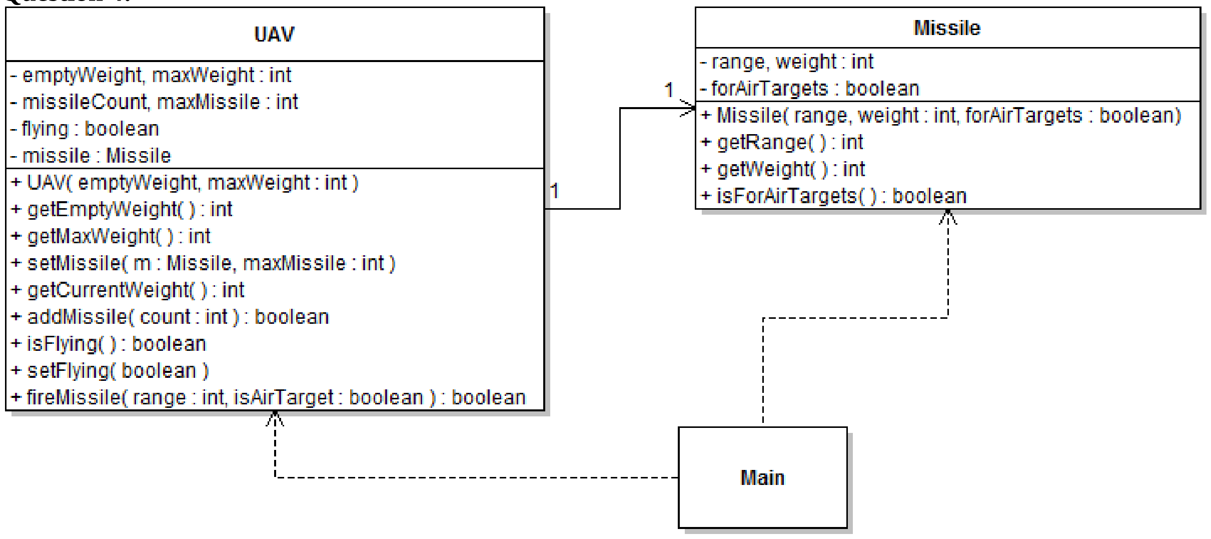
Question 2:

```java
public class UAV {
    private int emptyWeight, maxWeight;
    private Missile missile;
    private int missileCount, maxMissile;
    private boolean flying;

    public UAV(int emptyWeight, int maxWeight) {
        this.emptyWeight = emptyWeight;
        this.maxWeight = maxWeight;
    }
    public int getEmptyWeight() { return emptyWeight; }
    public int getMaxWeight() { return maxWeight; }
    public boolean isFlying() { return flying; }
    //public void setFlying(boolean flying) { this.flying = flying; }

    public void setMissile(Missile missile, int maxMissile) {
        this.missile = missile;
        this.maxMissile = maxMissile;
        missileCount = 0;
    }
    public int getCurrentWeight( ) {
        int result = emptyWeight;
        if( missile != null )
            result += missileCount * missile.getWeight();
        return result;
    }
    public boolean addMissile( int count ) {
        if( getCurrentWeight() + count * missile.getWeight() < maxWeight
                && missileCount + count <= maxMissile ) {
            missileCount += count;
            return true;
        }
        return false;
    }
    public boolean fireMissile( int range, boolean isAirTarget ) {
        if( missile != null && missileCount > 0
                && missile.getRange() > range && isFlying() &&
                missile.isForAirTargets() == isAirTarget ) {
            missileCount--;
            return true;
              }
        return false;
    }
}
```

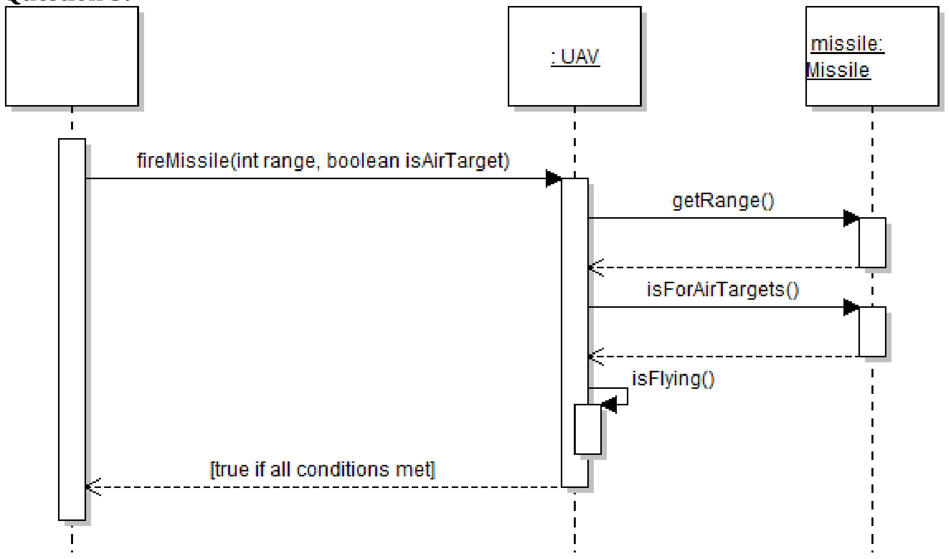
Question 3:

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Missile sidewinder = new Missile(1200, 100, true);
        UAV predator = new UAV(5000, 20000);
        Scanner in = new Scanner( System.in );
        System.out.print("How many missiles? ");
        int count = in.nextInt();
        predator.setMissile(sidewder, count);
        predator.addMissile(count);
        predator.setFlying(true);
        if(predator.fireMissile(100, true))
            System.out.println("Test is successful");
        else
            System.out.println("Test has failed");
        in.close();
    }
}
```

Question 4:


UAV
- emptyWeight, maxWeight : int
- missileCount, maxMissile : int
- flying : boolean
- missile : Missile
+ UAV(emptyWeight, maxWeight : int)
+ getEmptyWeight( ) : int
+ getMaxWeight( ) : int
+ setMissile(m : Missile, maxMissile : int)
+ getCurrentWeight( ) : int
+ addMissile(count : int) : boolean
+ isFlying( ) : boolean
+ setFlying(boolean)
+ fireMissile(range : int, isAirTarget : boolean) : boolean
1
1
Missile
- range, weight : int
- forAirTargets : boolean
+ Missile(range, weight : int, forAirTargets : boolean)
+ getRange( ) : int
+ getWeight( ) : int
+ isForAirTargets( ) : boolean
Main

Question 5:


: UAV
missile:
Missile
fireMissile(int range, boolean isAirTarget)
getRange()
isForAirTargets()
isFlying()
[true if all conditions met]

Topic: Abstract Classes

Soru 1: Bir programda 3 tür kişi oluşturulabilecektir: Çocuk, genç yetişkin ve yetişkin. Her kişinin adı olmak zorundadır ve değiştirilemez. Kişi bir çocuk ise şeker alabilir ama sigara alamaz. Kişi bir genç yetişkin ise şeker de sigara da alabilir ama sigara aldığında ekrana "Sigarayı bırakmalısın" öğüdü yazılmalıdır. Kişi yetişkinse şeker de sigara da alabilir ama şeker aldığında ekrana "şişmanlamamaya dikkat et" öğüdü yazılmalıdır.

Normal çözüm: Soyut sınıf kullanarak. UML şemalarının çizimi öğrenciye bırakılmıştır.

```java
public abstract class Person {
   private final String name;

   public Person(String name) { this.name = name; }
   public String getName() { return name; }
   public abstract void buyCandy();
   public abstract void buyCigarette();
}

public class Kid extends Person {
   public Kid(String name) {
      super(name);
   }
   public void buyCigarette() {
      System.out.println("A kid cannot buy cigarettes");
      return;
   }
   public void buyCandy() {
      System.out.println("Wow. So many candies!");
   }
}

public class YoungAdult extends Person {
   public YoungAdult(String name) {
      super(name);
   }
   public void buyCigarette() {
      System.out.println("Cigarettes are bought. Consider quitting.");
   }
   public void buyCandy() {
      System.out.println("Candies are bought.");
   }
}

public class Adult extends Person {
   public Adult(String name) {
      super(name);
   }
   public void buyCigarette() {
      System.out.println("Cigarettes are bought. Smoking Kills!");
   }
   public void buyCandy() {
      System.out.println("Candies are bought. Do not get fat.");
   }
}
```

```java
public class MainApp {
   public static void main( String[] args ) {
      Person[] people = new Person[3];
      people[0] = new Kid("Olcan Selçuk");
      people[1] = new YoungAdult("Fırat Ertemel");
      people[2] = new Adult("Yunus Emre Selçuk");
      for(Person person : people) {
         person.buyCandy();
         person.buyCigarette();
      }
   }
}
```

Aşağıdaki çözüm Strategy tasarım kalıbına göre arayüz kullanılarak yapılmıştır. UML şemalarının çizimi öğrenciye bırakılmıştır.

```java
public interface IAge {
   public void buyCigarette();
   public void buyCandy();
}
public class Kid implements IAge {
   public void buyCigarette() {
      System.out.println("A kid cannot buy cigarettes");
      return;
   }
   public void buyCandy() {
      System.out.println("Wow. So many candies!");
   }
}

public class YoungAdult implements IAge {
   @Override
   public void buyCigarette() {
      System.out.println("Cigarettes are bought. Consider quitting.");
   }
   @Override
   public void buyCandy() {
      System.out.println("Candies are bought.");
   }
}
public class Adult implements IAge {
   @Override
   public void buyCigarette() {
      System.out.println("Cigarettes are bought. Smoking Kills!");
   }
   @Override
   public void buyCandy() {
      System.out.println("Candies are bought. Do not get fat.");
   }
}
```

```java
public class Person {
   private String name;
   private IAge age;

   public Person(String name, IAge age) {
      this.name = name;
      this.age = age;
   }
   public String getName() {
      return name;
   }
   public void buyCandies() {
      age.buyCandy();
   }
   public void buyCigars() {
      age.buyCigarette();
   }
   public static void main( String[] args ) {
      Person[] people = new Person[3];
      people[0] = new Person("Olcan Selçuk", new Kid());
      people[1] = new Person("Fırat Ertemel", new YoungAdult());
      people[2] = new Person("Yunus Emre Selçuk", new Adult());
      for(Person person : people) {
         person.buyCandies();
         person.buyCigars();
      }
   }
}
```