Crag
Crag is a dice game involving three dice. In a simple application, write the Java code that uses the Die class to build the three dice, roll them and print their face up values.

The algorithm is simply:

```
// build die
roll die
output the die's face up value
```

These are repeated three times for each of the three dice. Here’s the application:

```
public class Crag {
    public static void main( String [] args ) {
        // build 3 dice
        Die die1 = new Die( );
        Die die2 = new Die( );
        Die die3 = new Die( );
        // roll them
        die1.roll( );
        die2.roll( );
        die3.roll( );
        // print them
        System.out.println( "Die 1 shows " + die1.faceUp );
        System.out.println( "Die 2 shows " + die2.faceUp );
        System.out.println( "Die 3 shows " + die3.faceUp );
    }
}
```
Crag Scoring
Crag players score points by obtaining certain combinations on the three dice. For example, this table shows three scoring combinations.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crag</td>
<td>A pair plus another totaling 13</td>
<td>5 5 3</td>
<td>50</td>
</tr>
<tr>
<td>Thirteen</td>
<td>Any 3 dice totaling 13</td>
<td>6 5 2</td>
<td>26</td>
</tr>
<tr>
<td>Three of a kind</td>
<td>Any combination of three equal dice</td>
<td>2 2 2</td>
<td>25</td>
</tr>
</tbody>
</table>

Extend the Crag dice example so that, after rolling the three dice, the application determines whether or not any of these three combinations are obtained.

The initial algorithm rolls the dice and checks for one of these combinations:

```
Crag Scoring Algorithm (1st Draft)
build 3 dice
roll the dice
if a "Crag" is obtained then
    score = 50
else if "Thirteen" is obtained then
    score = 26
else if "Three of a kind" is obtained then
    score = 25
else
    score = 0
end if
```
We can simplify the coding considerably, however, by realizing that “Crag” is a special case of “Thirteen.” If we know that the dice add up to thirteen then we can check for the “Crag” by determining if any two of them are equal. The revised algorithm is shown below.

The completed program is shown below. It makes use of the following Java constructs covered in subsequent topics: the `if-else`, discussed in *The Java if-else Statement and Cascading If-Else*; and the operators `&&` and `||`, discussed in *Logical Operators*.

```java
public class CragScore {

    public static void main( String [] args ) {
        int score;
        // build 3 dice
        Die die1 = new Die( );
        Die die2 = new Die( );
        Die die3 = new Die( );
        // roll them
        die1.roll( );
        die2.roll( );
        die3.roll( );
        // print them
```
15 System.out.println( "Die 1 shows " + die1.faceUp );
16 System.out.println( "Die 2 shows " + die2.faceUp );
17 System.out.println( "Die 3 shows " + die3.faceUp );
18 // determine the score
19 if ( die1.faceUp + die2.faceUp + die3.faceUp == 13 )
20   if ( die1.faceUp == die2.faceUp
21       || die1.faceUp == die3.faceUp
22       || die2.faceUp == die3.faceUp )
23     // "Crag"
24     score = 50;
25 else
26   // ordinary "Thirteen"
27     score = 26;
28 else if ( die1.faceUp == die2.faceUp
29       && die2.faceUp == die3.faceUp )
30   // "Three of a kind"
31     score = 25;
32 else
33     score = 0;
34 System.out.println( "Score is " + score );
35 }
36 }

Craps
Gambling experts consider Craps to be one of the best casino games because the house edge is a small 1.41%, meaning that after playing a long time making many bets, on the average you’ll lose only 1.41¢ for every dollar that you bet.

For this example, we create a Java application that simulates the Craps game, tossing two dice and determining if the toss wins or loses. If you know how to play Craps then you should understand the algorithm below. If you don’t know how to play, look on the Internet for an explanation. Michael Bluejay’s web site\(^1\) has a particularly lucid explanation and includes a simulator that allows you to practice the game.

The completed program is shown below. It uses the Die class to model the two dice. To make the program more usable, it outputs the result of each dice roll and the sum of the dice. The big conditional construct in the algorithm is code using a Java switch statement, which is discussed in the subsequent topic The Java switch Statement. The loop is coded with a do-while, which is discussed in The Java do-while Statement.

```java
public class Craps {
    public static void main( String [] args ) {
        // declare data
        Die die1 = new Die( );
        Die die2 = new Die( );
        int sum, thePoint;
        die1.roll( );     // roll the dice
        die2.roll( );     // and sum them
        sum = die1.faceUp + die2.faceUp;
```
System.out.print( "Come-out roll: " + die1.faceUp );
System.out.print( " + " + die2.faceUp );
System.out.println( " = " + sum );
// evaluate the come-out roll
switch ( sum ) {
    case 7: case 11: // player wins
        System.out.println( "Player wins" );
        break;
    case 2: case 3: case 12: // player loses
        System.out.println( "Player loses" );
        break;
    default: // set "the point" and continue
        thePoint = sum;
        System.out.println( "The point = " + thePoint );
        do {
            // roll until 7 or thePoint
            die1.roll( );
            die2.roll( );
            sum = die1.faceUp + die2.faceUp;
            System.out.print( "Roll: " + die1.faceUp );
            System.out.print( " + " + die2.faceUp );
            System.out.println( " = " + sum );
        } while ( sum != 7 && sum != thePoint );
        if ( sum == 7 )
            System.out.println( "Player loses" );
        else
            System.out.println( "Player wins" );
        break;
} // end switch
} // end switch