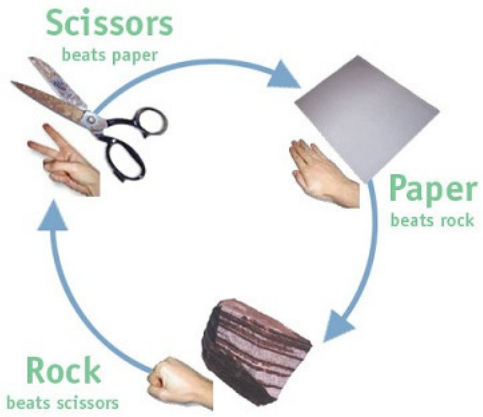
ROCK, PAPER, SCISSORS

Rock, Paper, Scissors is a two-player game where each player simultaneously chooses one of *Rock*, *Paper* or *Scissors*. The winner is determined by the following rules:



- Scissors cuts Paper
- Paper covers Rock
- Rock smashes Scissors

We want to implement a Java application that plays the game with the program being one player and the user the other. After each play, the program must award the winner 1 point (no point is given in a tie) and print the players' scores. Play continues until one player reaches 5 points. If the user reaches 5, print congratulations; if the program reaches 5, print a gloating, nasty message. In both cases, the program must ask the user if he or she wishes to play again and respond appropriately. Also, the program must allow the user to escape the program by entering a special signal instead of rock, paper or scissors. The user's input must be case-insensitive and the program must respond with an appropriate message if the input is incorrect.

Algorithm #1

```
print welcome message
do
    play rock, paper, scissors repeatedly until one of the players reaches 5 points
    if it's me then
        gloat
    else
        congratulate user
    end if
    ask user if he/she wants another game
while user wants another game
```

Algorithm #2

```
print welcome message
do
    myScore = yourScore = 0
    while myScore != 0 AND yourScore != 0 do
        play rock, paper, scissors
        if I win then
            add 1 to myScore
        else
            add 1 to yourScore
        end if
    end while
    if myScore == 5 then
        gloat
    else
        congratulate user
    end if
    ask user if he/she wants another game
```

Algorithm #3

```
print welcome message
do
    myScore = yourScore = 0
    while myScore != 0 and yourScore != 0 do
        myChoice = ROCK, PAPER or SCISSORS randomly
        input yourChoice
        if yourChoice is invalid then
            cycle to try again
        else if yourChoice is to quit then
            quit
        end if
        print the choices
        if I win then
            print "I won"
            add 1 to myScore
        else
            print "You won"
            add 1 to yourScore
        end if
    end while
    if myScore == 5 then
        gloat
    else
        congratulate user
    end if
    ask user if he/she wants another game
while user wants another game
```

Here's the Java application:

```
import java.util.Scanner;

public class RockPaperScissors
{
   public static void main ( String [] args )
   {
      // data -- "me" = program; "you" = user
      int myScore, yourScore;   // scores
      int myChoice, yourChoice; // int rock, paper, scissors
      String myWord, yourWord;  // word rock, paper, scissors
      // int values that represent rock, paper, scissors
      final int ROCK     = 0; // improves program readability
      final int PAPER    = 1;
      final int SCISSORS = 2;
      // user response to "Play Again?" query
      String response;
      // print introduction
      System.out.println
         ( "\nLet's play \"Rock, Paper, Scissors!\"" );
      System.out.println( "The first to get 5 points wins." );
      // build Scanner object for user input
      Scanner input = new Scanner( System.in );
      do { // until user says to quit
         // initialize scores
         myScore = yourScore = 0;
         // play until one of us wins
         while ( myScore < 5 && yourScore < 5 )
         {
            myChoice = (int)(Math.random( ) * 3); // I choose
            // turn my choice into a word suitable for printing
            myWord = "UNKNOWN"; // make compiler happy
            switch ( myChoice )
            {
               case ROCK:
                  myWord = "ROCK";
                  break;
              case PAPER:
```

```
            myWord = "PAPER";
            break;
         case SCISSORS:
            myWord = "SCISSORS";
            break;
      }  // end switch
      // now get your choice as a word
      System.out.print( "\nRock, Paper or Scissors? " );
      yourWord = input.next( );
      // convert your choice to upper-case
      yourWord = yourWord.toUpperCase( );
      yourChoice = -1; // make compiler happy
      switch ( yourWord ) // what did you type?
      {
         case "QUIT": // user wants to escape
            System.out.println( "\nGood bye." );
            System.exit( 0 );
            break;
         case "ROCK": // turn to int for processing
            yourChoice = ROCK;
            break;
         case "PAPER":
            yourChoice = PAPER;
            break;
         case "SCISSORS":
            yourChoice = SCISSORS;
            break;
         default: // error in user input
            System.out.println( "\nYou mistyped your response." );
            continue; // force next loop cycle
      }  // end switch
      // print choices
      System.out.print( "\nYou chose " + yourWord );
      System.out.print( ". I chose " + myWord );
      // who won?
      if ( myChoice == (yourChoice+1)%3 ) // I did
      {  // myChoice is 1 larger (3-hour clock arithmetic)
         System.out.println( ". My point!" );
         myScore++;
```

```
            }
           else if ( yourChoice == (myChoice+1)%3 ) // You did
            {  // yourChoice is 1 larger (clock arithmetic)
               System.out.println( ". Your point!" );
               yourScore++;
            }
            else if ( yourChoice == myChoice ) // We tie
            {  // our choices are the same
               System.out.println( ". A tie!" );
            }  // end who won
            // print scores
            System.out.println( "\nYour score: " + yourScore );
            System.out.println(   "My score:   " + myScore );
         }  // end while
         // assess winner
         if ( yourScore == 5 )
            System.out.println( "\nCongratulations!" );
         else
            System.out.println( "\nHa Ha! Loser!" );
         System.out.print( "\nPlay again (y/n)? " );
         response = input.next( );
      }  while ( response.equalsIgnoreCase( "Y" ) );
   }  // end main method
}
```

Problem Solving Exercises

For each of the following exercises, invent an algorithm that solves the problem. Code your algorithm into a Java application and deploy it. Follow all programming conventions and create readable output through the use of formatter objects or string formatting methods. Experiment with different styles of interaction – through the standard input and output objects and through dialog boxes.

1. Write a Java application that plays a simple guessing game. The program chooses an integer at random in the range 1 to 1000 and gives the user 10 chances to guess it. After each guess tell the user his or her guess is "correct," "too high" or "too low." Make the game output easy for the user to follow. Here is a sample interaction (shown with text typed by the user underlined and text output by the program not underlined):

```
Let's play a guessing game!
I've thought of a number from 1 to 1000.
Can you guess it within 10 attempts?

Guess #1? 500
Too low! Try again.

Guess #2? 750
Too high! Try again.

Guess #3? 625
Too low! Try again.

Guess #4? 688
You got it!
```

If the user guesses correctly by the 10th chance, congratulate him or her. If, after the 10th guess, the user still hasn't guessed the number, print a nasty message. In either case, ask the user if he or she wants to play again and cycle to play again if the answer is "y" or "Y". For example:

```
Guess #4? 688
You got it!

Congrats!
Do you want to play again (y/n)? y
. . .
```

2. Turn the previous problem's guessing game around so that the program guesses the numbers and the user indicates if the guess is "correct," "too high" or "too low." Each guess must be randomly chosen within the remaining possibilities. For example, if the program has previously been told that 100 is "too low" and 325 is "too high" then it must randomly choose a number in the range 101 to 324. Make the game output easy for the user to follow. Here is a sample interaction (shown with text typed by the user underlined and text output by the program not underlined):

```
Let's play a guessing game!
Think of a number from 1 to 1000.
I'll try to guess it in 10 guesses.

Guess #1: 500
Too low, too high or correct? too high

Guess #2: 150
Too low, too high or correct? too low

Guess #3: 225
Too low, too high or correct? correct
```

All user input must be case-insensitive and the program must respond with appropriate messages if the input is invalid. For example:

```
Too low, too high or correct? to hi
Sorry, your input is invalid.
Too low, too high or correct? Too High
```

The game ends when the program guesses correctly or 10 guesses have been made, Print an appropriate message "good for me" or "darn it" message and then ask the user if he or she wants to play again and cycle to play again if the answer is "Yes."

The program must not be confused or give a run-time error if the player is careless or dishonest. For example, suppose the program guesses 100 and the player responds "too low." Later the program guesses 101 and the player responds "too high." The player is wrong and the program must say so and halt the game.

3. Write a Java application that allows a user to play the dice game Craps. For this program, you need only allow Pass Line bets. If you don't know the game, there are many good Internet sites that explain it and allow you to practice it free of charge. To find them Google "craps." The site http://vegasclick.com/games/craps.html is particularly well written.

Your program must first welcome the player and input the amount of money in his or her bankroll. Each game transaction must proceed as follows:

Main Algorithm
Repeat:
    Ask the user to place a bet and subtract it from the bankroll.
    If the bet is 0, the player is signaling that he or she wants to quit.
    If the player's bankroll is 0, then quit.
    Play the game.
    If the player wins, then add the winnings to the bankroll

Craps Game Algorithm:
Roll the two dice.
If the sum is 2, 3 or 12, you lose.
If the sum is 7 or 11, you win even money (i.e. equal to your bet).
If the sum is anything else:
    The sum becomes the Point.
    Repeat:
        Roll the two dice.
        If the sum is 7, you lose.
        If the sum equals the Point, you win even money.
If the sum is anything else, continue with the next roll.

Your program must output the bankroll before asking for the bet. Make the game output easy for the user to follow. Here is a sample (shown with text typed by the user underlined and text output by the program not underlined):

```
Your bankroll is $500
Place your bet: 20
Come-out roll: 5 + 5 = 10
The point = 10
Roll: 6 + 6 = 12
Roll: 5 + 5 = 10
You win.

Your bankroll is $520
```

```
Place your bet: 15
Come-out roll: 1 + 6 = 7
You win on come-out roll.

Your bankroll is $535
Place your bet: 30
Come-out roll: 1 + 3 = 4
The point = 4
Roll: 2 + 1 = 3
Roll: 3 + 3 = 6
Roll: 4 + 4 = 8
Roll: 5 + 6 = 11
Roll: 1 + 6 = 7
You lose.

Your bankroll is $505
Place your bet:
. . .
```