Helper Methods

Splitting Up Complicated Code

- The program to run a series of guess games is quite complicated.
- We made the design easier by splitting it into two parts.
  - The games sequence.
  - An individual game.
- Helper methods let us write the code in two or more parts as well.

A Single Game

- We can write a helper method to play a single game.
- The `main` method can then use this helper method to run a series of games.
- We can test the single game method first.
- Then test the series when the single game code works.

Inputs

- Helper methods need inputs to get started.
- Play a game needs:
  - A `Console` object to communicate with the user.
  - A `Random` object to choose random numbers.
- This information is provided by the `main` method.
- Inputs are called ***parameters***.

Outputs

- The play a game method must tell the `main` method how many guesses the user needed.
- This information is given back to the `main` method.
- `main` can use it to calculate the average number of guesses needed.

private static

- Helper methods are `private`.
- This means that they can only be called by `main` or another helper method.
  - More about `public` and `private` later.
- The `main` method is `static` and so all helper methods must be `static` as well.
  - More on `static` later.

The Play a Game Method

```
private static int playGame(Console con, Random rand)
{
      // choose a random number between 0 and 9
   int num = Math.abs(rand.nextInt()) % 10;
   System.err.println(num); // for testing

      // variable to count number of attempts
   int count = 0;

      // loop until user gets it right
   for (;;)   // break out loop
   {
```

Play a Game (2)

```
                // get the user's guess
            con.print("Guess a number: ");
            int guess = con.readInt();
            count++;

                // print how they did
            if (guess == num)
            {
                con.println("CORRECT!!!, you took "
                        + count + " guesses");
                return count;
            }
            else if (guess > num)
                con.println("Too high");
            else
                con.println("Too low");
    }
}
```

Method Header

- Tells how the method will be used.

```
private static int playGame(Console con, Random rand)
```

- The method name: `playGame`
- The parameters are like variable definitions
  - `(Console con, Random rand)`
- The return type: `int`
- Visibility: `private` (more later)
- Lifetime: `static` (more later)
- The rest is the ***method body***.

Parameters

- Parameters are inputs to the method.
- They are variables with initial values.
  - `main` will provide these initial values.
- They can be used anywhere in the method.
- They come straight after the method name.
  - `playGame(Console con, Random rand)`
- There are 2 inputs, a `Console` called `con` and a `Random` called `rand`.

Return Value

- In Java a method can only return one value.
- In this case we want to return an integer value.
  - The number of guesses.
- The type of return value comes before the method name.
  - `int playGame`

return Statement

- A `return` statement sends a value back to `main`.
  - `return count;`
- The value returned must be the same type as that specified in the method header.
- There can be many `return` statements in a method.
- Control is transferred back to the calling method when a return is encountered.

Running a Method

- This is also called ***calling*** a method.
- `main` must give `playGame` the information it needs
  - The two parameters `con` and `rand`.
  - They must be given in the order that `playGame` expects them.
- `main` must deal with the value returned.
  - The number of guesses.

Calling playGame

```
    // create helper objects
Console con = new Console("Guess");
Random rand = new Random();

    // remember total score and number of games
int totalScore = 0, numGames = 0;

for (;;)   // repeat games loop
{
    // call playGame helper method
    totalScore += playGame(con, rand);
    numGames++;
```

Calling playGame (2)

```
            // see if should go round again
        con.print("Do you want another go? ");
        String reply = con.readWord();
        if (reply.equals("no"))
            break;
    }

        // print out average score
    double average = (double)totalScore / numGames;
    con.print("Thank you for playing, your average was");
        // average needs precision
    con.println(String.format(“%6.2f”, average));
```

Calling playGame explained

- Here is the line that calls `playGame`.
  - `totalScore += playGame(con, rand);`
- The two parameters `con` and `rand` are helper objects.
  - They are created in `main`.
  - They are given to `playGame` to use.
- The `int` returned by `playGame` is used by adding it to `totalScore`.

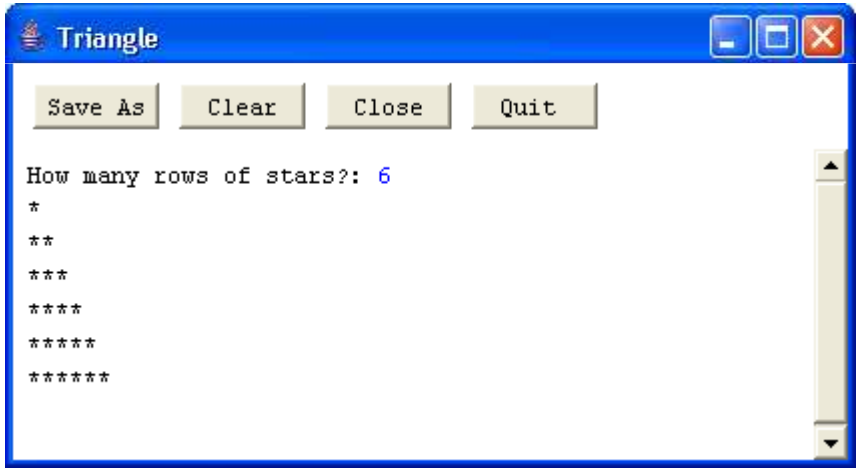
Layout Of The Program

- Here is how the main and helper methods are written in the program.

```
import FormatIO.*;
import java.util.*;
import java.lang.*;
public class Ex3 {
public static void main(String[] arg)
{
   // body of main
}
private static int playGame(Console con, Random rand)
{
   // body of playGame
}
} // end of class Ex3
```

Drawing a Triangle

- This example aims to draw a triangle made of stars, as the sample output shows.



Overall Design

- Get number of rows from user
- Loop with i from 1 to number of rows
  - Draw a row with i stars

Draw A Row

- Input: Console, number of stars in the row.
- Output: None.

- Loop for number of stars
  - Draw a *
- Draw a newline.

drawRow Helper Method

- Header
  - Name: `drawRow`
  - Parameters `(Console con, int nStars)`
  - Return type: `void` (no output).

```
private static void drawRow(Console con, int nStars)
{
   for (int i = 1; i <= nStars; i++)
       con.print("*");
   con.println();
}
```

main Method

```
public static void main(String[] arg)
{
       // get info from user
   Console con = new Console("Triangle");
   con.print("How many rows of stars?: ");
   int nRows = con.readInt();

       // loop through rows
   for (int i = 1; i <= nRows; i++)
       drawRow(con, i);
}
```

Things To Note

- The variable name `i` is used in both methods.
  - It is the name of two different variables.
  - They both have different scope.
  - The scopes do not overlap.
- The parameter is called `nStars` in `drawRow`.
  - It is initialised with `i` in `main`.
  - The names do not have to be the same.

A Fixed Size Triangle

- This clumsy example shows repeated calls to the `drawRow` method.
- It will draw a triangle with 4 rows.

```
public static void main(String[] arg)
{
  Console con = new Console("Triangle");
  drawRow(con, 1);
  drawRow(con, 2);
  drawRow(con, 3);
  drawRow(con, 4);
}
```