Transferring data between activities:

<http://www.101apps.co.za/index.php/articles/passing-data-between-activities.html>

Refactoring:

<https://en.wikipedia.org/wiki/Code_refactoring>

<http://refactoring.com/>

Options for Refactoring:

* Rename an identifier
* Rename a method (The name of a method does not reveal its purpose. Change the name of the method.)
* Encapsulate field (there is a public field, make it private and provide accessors)
* Hide method (a method is not used by any other class, make the method private)
* Introduce null object (you have repeated checks for a null value. Replace the null value with a null object)
* Lazily initialized attribute (an attribute takes time to initialize but is only accessed rarely, initialize when it’s first used)
* Move field (a field is, or will be, used by another class more than the class on which it is defined. Create a new field in the target class, and change all its users)
* Move method (A method is, or will be, using or used by more features of another class than the class on which it is defined. Create a new method with a similar body in the class it uses most. Either turn the old method into a simple delegation, or remove it altogether.)
* Preserve whole object (You are getting several values from an object and passing these values as parameters in a method call. Send the whole object instead.)
* Remove assignments to parameters (The code assigns to a parameter. Use a temporary variable instead.)
* Remove parameter (A parameter is no longer used by the method body. Remove it)
* Replace data value with object (You have a data item that needs additional data or behavior. Turn the data item into an object)
* Replace error code with exception (A method returns a special code to indicate an error. Throw an exception instead)
* Replace loop with collection closure method (You are processing the elements of a collection in a loop. Replace the loop with a collection closure method)
* Replace method with method object (You have a long method that uses local variables in such a way that you cannot apply. Turn the method into its own object so that all the local variables become fields on that object. You can then decompose the method into other methods on the same object.)