Happy Number:

```
import java.util.HashSet;
import java.util.Set;

class Solution {

    public boolean isHappy(int n) {
        Set<Integer> seen = new HashSet<>();

        while (n != 1 && !seen.contains(n)) {
            seen.add(n);
            n = getSumOfSquares(n);
        }

        return n == 1;
    }

    private int getSumOfSquares(int num) {
        int sum = 0;
        while (num > 0) {
            int digit = num % 10;
            sum += digit * digit;
            num /= 10;
        }
        return sum;
    }

    public static void main(String[] args) {
        Solution solution = new Solution();
        int n1 = 19;
        System.out.println(solution.isHappy(n1));

        int n2 = 2;
        System.out.println(solution.isHappy(n2));
    }
}
```

Palindrome Number:

```
class Solution {
    public boolean isPalindrome(int x) {
        if (x < 0) {
            return false;
        }

        int original = x;
        int reversed = 0;

        while (x != 0) {
            int digit = x % 10;
            reversed = reversed * 10 + digit;
            x /= 10;
        }

        return original == reversed;
    }

    public static void main(String[] args) {
        Solution solution = new Solution();
        System.out.println(solution.isPalindrome(121));
        System.out.println(solution.isPalindrome(-121));
        System.out.println(solution.isPalindrome(10));
    }
}

class ListNode {
    int val;
    ListNode next;

    ListNode() {}
    ListNode(int val) { this.val = val; }
    ListNode(int val, ListNode next) { this.val = val; this.next = next; }

    public static ListNode deserialize(String data) {
        if (data == null || data.isEmpty()) {
            return null;
        }

        data = data.replace("[", "").replace("]", "");

        String[] values = data.split(",");
        ListNode dummyHead = new ListNode(0);
        ListNode current = dummyHead;

        for (String value : values) {
            if (!value.trim().isEmpty()) {
                current.next = new ListNode(Integer.parseInt(value.trim()));
                current = current.next;
            }
        }
        return dummyHead.next;
    }

    public static void printList(ListNode node) {
        while (node != null) {
            System.out.print(node.val + " ");
            node = node.next;
        }
        System.out.println();
    }
}
```

Add Two Numbers

```
class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
        ListNode dummyHead = new ListNode(0);
        ListNode current = dummyHead;
        int carry = 0;

        while (l1 != null || l2 != null || carry != 0) {
            int sum = carry;

            if (l1 != null) {
                sum += l1.val;
                l1 = l1.next;
            }

            if (l2 != null) {
                sum += l2.val;
                l2 = l2.next;
            }

            carry = sum / 10;
            int digit = sum % 10;

            current.next = new ListNode(digit);
            current = current.next;
        }



        return dummyHead.next;
    }

    public static void main(String[] args) {
        ListNode l1 = ListNode.deserialize("[2,4,3]");
        ListNode l2 = ListNode.deserialize("[5,6,4]");

        Solution solution = new Solution();
        ListNode result = solution.addTwoNumbers(l1, l2);

        ListNode.printList(result);
    }
}
```

Two Sum

```
import java.util.HashMap;
import java.util.Map;

public class Solution {
```

```
public int[] twoSum(int[] nums, int target) {
    Map<Integer, Integer> map = new HashMap<>();

    for (int i = 0; i < nums.length; i++) {
        int complement = target - nums[i];

        if (map.containsKey(complement)) {
            return new int[] { map.get(complement), i };
        }

        map.put(nums[i], i);
    }

    throw new IllegalArgumentException("No two sum solution");
}

public static void main(String[] args) {
    Solution solution = new Solution();

    int[] nums1 = {2, 7, 11, 15};
    int target1 = 9;
    int[] result1 = solution.twoSum(nums1, target1);
    System.out.println("Indices: [" + result1[0] + "," + result1[1] + "]");

    int[] nums2 = {3, 2, 4};
    int target2 = 6;
    int[] result2 = solution.twoSum(nums2, target2);
    System.out.println("Indices: [" + result2[0] + "," + result2[1] + "]");

    int[] nums3 = {3, 3};
    int target3 = 6;
```

```
        int[] result3 = solution.twoSum(nums3, target3);
        System.out.println("Indices: [" + result3[0] + "," + result3[1] + "]");
    }
}
```

Same Tree

```
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode(int val) {
        this.val = val;
    }
}

public class Solution {
    public boolean isSameTree(TreeNode p, TreeNode q) {
        if (p == null && q == null) {
            return true;
        }
        if (p == null || q == null) {
            return false;
        }
        return (p.val == q.val)
                && isSameTree(p.left, q.left)
                && isSameTree(p.right, q.right);
    }

    public static void main(String[] args) {
        TreeNode p1 = new TreeNode(1);
        p1.left = new TreeNode(2);
        p1.right = new TreeNode(3);

        TreeNode q1 = new TreeNode(1);
        q1.left = new TreeNode(2);
        q1.right = new TreeNode(3);

        Solution solution = new Solution();
        System.out.println(solution.isSameTree(p1, q1));

        TreeNode p2 = new TreeNode(1);
        p2.left = new TreeNode(2);

        TreeNode q2 = new TreeNode(1);
        q2.right = new TreeNode(2);

        System.out.println(solution.isSameTree(p2, q2));
    }
}
```

Merge Sorted Array

```
import java.util.Arrays;

public class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        int i = m - 1;
        int j = n - 1;
        int k = m + n - 1;

        while (i >= 0 && j >= 0) {
            if (nums1[i] > nums2[j]) {
                nums1[k] = nums1[i];
                i--;
            } else {
                nums1[k] = nums2[j];
                j--;
            }
            k--;
        }

        while (j >= 0) {
            nums1[k] = nums2[j];
            j--;
            k--;
        }
    }

    public static void main(String[] args) {
        Solution solution = new Solution();

        int[] nums1 = {1, 2, 3, 0, 0, 0};
        int m = 3;
        int[] nums2 = {2, 5, 6};
        int n = 3;
        solution.merge(nums1, m, nums2, n);
        System.out.println(Arrays.toString(nums1));

        int[] nums1_2 = {1, 0};
        int m2 = 1;
        int[] nums2_2 = {};
        int n2 = 0;
        solution.merge(nums1_2, m2, nums2_2, n2);
        System.out.println(Arrays.toString(nums1_2));

        int[] nums1_3 = {0};
        int m3 = 0;
        int[] nums2_3 = {1};
        int n3 = 1;
        solution.merge(nums1_3, m3, nums2_3, n3);
        System.out.println(Arrays.toString(nums1_3));
    }
}
```

Reverse Integer:

```
public class Solution {
    public int reverse(int x) {
        int reversed = 0;
        while (x != 0) {
            int digit = x % 10;
            x /= 10;

            if (reversed > Integer.MAX_VALUE / 10 || (reversed == Integer.MAX_VALUE / 10 && digit > 7))
{
                return 0;
            }
            if (reversed < Integer.MIN_VALUE / 10 || (reversed == Integer.MIN_VALUE / 10 && digit < -8)) {
                return 0;
            }

            reversed = reversed * 10 + digit;
        }
        return reversed;
    }


    public static void main(String[] args) {
        Solution solution = new Solution();



    }
}
```