# Chapter 14-3

1. **public** **int** makeingList(List<Integer> list){

**int** count=0;

**for** (**int** i : list)

count+= i;

**return** count;

}

1. List<String> list1 = (List<String>) **new** ArrayList();

**for** (String i: list1)

System.***out***.println(i);

## Projects

//Project 14-1

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.PrintWriter;

**import** java.util.\*;

**import** javax.swing.JOptionPane;

**public** **class** project141 {

**public** **static** **void** main(String[] args) **throws** FileNotFoundException {

String inputstr = JOptionPane.*showInputDialog*("enter file name(include file type like .txt)","number.txt");

Scanner file = **new** Scanner(**new** File(inputstr));

List<String> lst = **new** ArrayList<String>();

**while** (file.hasNext())

{

lst.add((file.next()).toUpperCase());

}

String outputstr = JOptionPane.*showInputDialog*("enter file name for output (include file type like .txt)","number.txt");

PrintWriter writer = **new** PrintWriter(**new** File(outputstr));

**for** (**int** i = 0; i < lst.size(); i++)

writer.println(lst.get(i));

}// end of main

}// end of project 14-1

//Project 14-5

**import** java.util.\*;

**public** **class** War {

**public** **static** **void** main(String[] args){

//how do I divide the deck into 2 decks to make two players

List<Card> player1Unplay = **new** ArrayList<Card>();

List<Card> player2Unplay = **new** ArrayList<Card>();

List<Card> war1 = **new** ArrayList<Card>();

List<Card> war2 = **new** ArrayList<Card>();

List<Card> winPile1 = **new** ArrayList<Card>();

List<Card> winPile2 = **new** ArrayList<Card>();

Deck deck = **new** Deck();

deck.shuffle();

//making a deck cards currently being played

**for** (**int** i = 0; i < deck.size()/2; i++)

{

player2Unplay.add(deck.deal());

player1Unplay.add(deck.deal());

}

**int** war=0, i =0;

**while**(!player1Unplay.isEmpty() && !player2Unplay.isEmpty())

{

war1.add(player1Unplay.remove(i));

war2.add(player2Unplay.remove(i));

System.out.print("\n" + "Player1 has put down " + war1 +"\n" + "Player2 has put down " + war2 +"\n" );

**if** (war1.get(war).compareTo(war2.get(war)) > 0)

{

**for** (**int** j = war; j >= 0; j--)

{

winPile1.add(war1.remove(j));

winPile1.add(war2.remove(j));

}

}

**else** **if** (war1.get(war).compareTo(war2.get(war)) < 0)

{

**for** (**int** k = war; k >= 0; k--)

{

winPile2.add(war1.remove(k));

winPile2.add(war2.remove(k));

}

war = 0;

}

**else**

{

i++;

war++;

}

}// end of loop

// winner!

**if** (winPile1.size() >winPile2.size())

System.out.println("Player 1 wins");

**else** **if** (winPile1.size() <winPile2.size())

System.out.println("Player two wins");

}// end of the main

}

# Unit Review pg. 483

1. F
2. F
3. T
4. T
5. T
6. T
7. T
8. Array
9. Index or subscript
10. Parallel
11. Logical
12. Abstract Classes
13. Inheritance
14. Polymorphic message

## Unit Review pg.484

1. A. int[] intNumbers = new int[5];

B. int[] realNumbers = new int[100];

C. Boolean[] bools = new Boolean[10];

D. String[] words = new String[20];

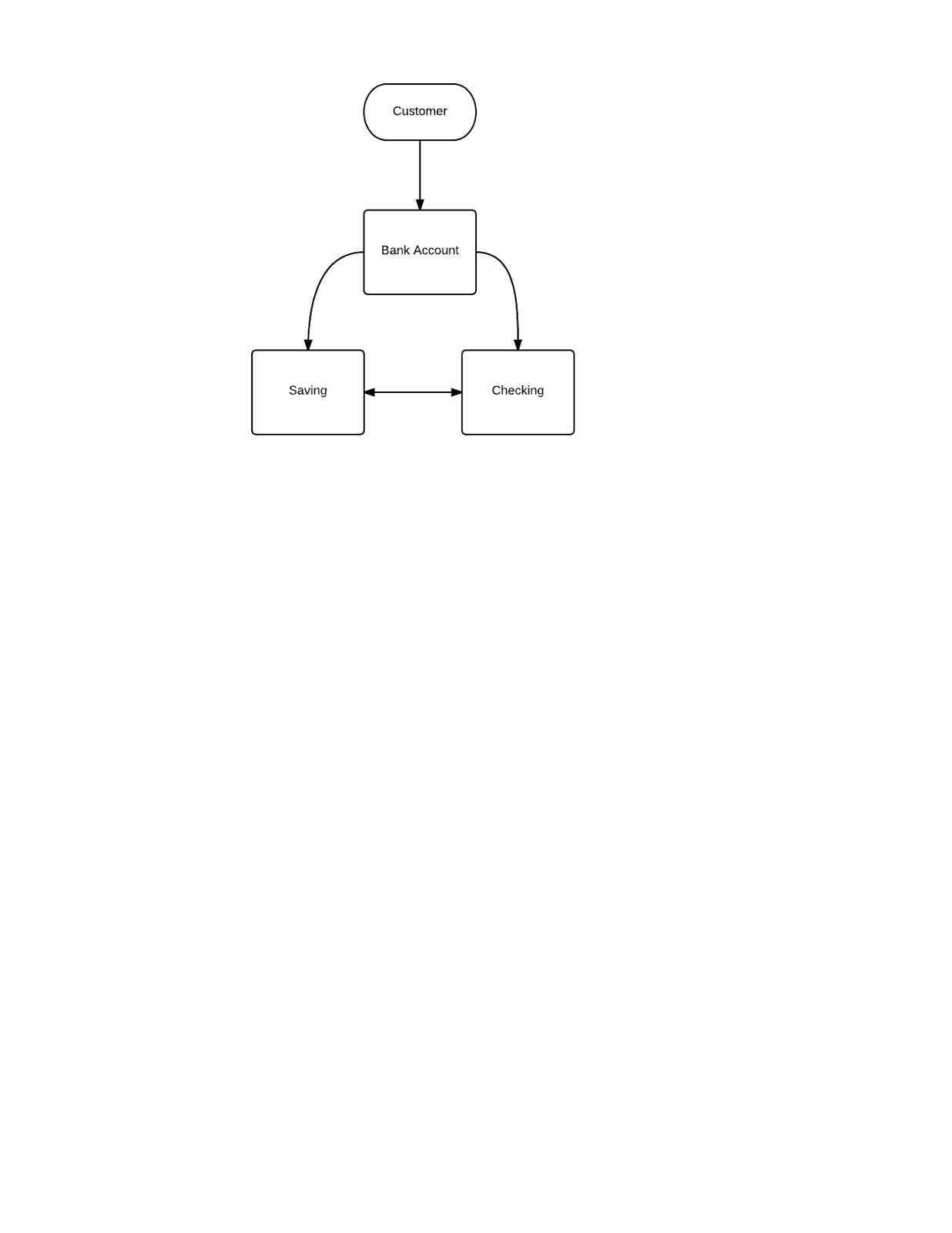
1. Int[] Num = new int[10];

For (int I = 0; I < num.length; i++)

Num[i] = I;

1. Int[] Num = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
2. Extend and implement
3. Sometimes, you want to have variables that are common to all objects. Every instance of the class shares a class variable, which is in one fixed location in memory. Any object can change the value of a class variable, but class variables can also be manipulated without creating an instance of the class.

## Critical Thinking pg. 485



## Critical Thinking p. 581

Arrays get better performance but ArrayList have extra functionality and arraylist do not have a set size.

## Ch 14 Review Questions p. 578-579

1. String and List
2. java.util.ArrayList and java.util.LinkedList
3. LinkedList
4. Java.util.\*