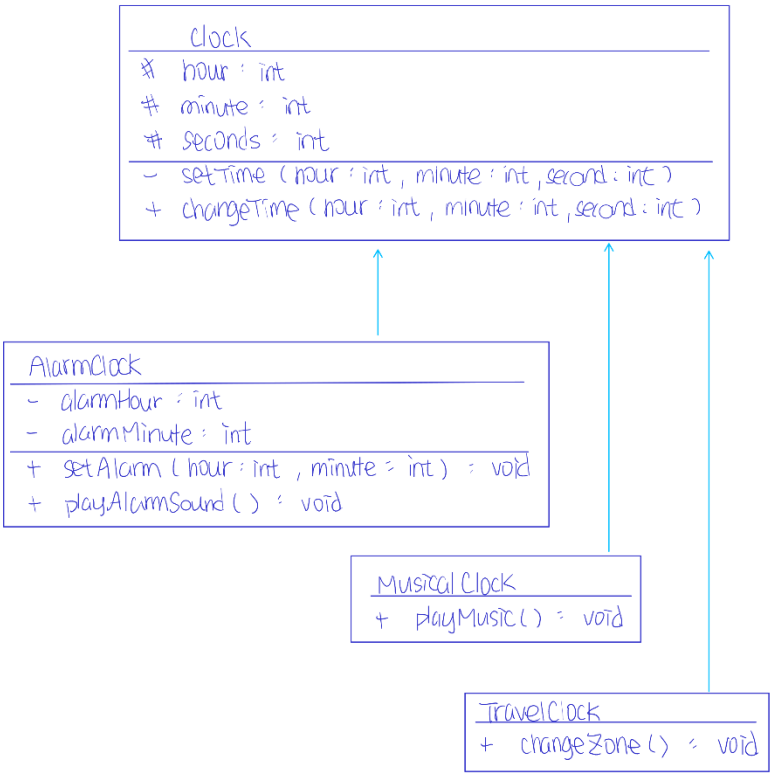
CSCI 145 Homework 2

Due Wednesday, 5/24/2023

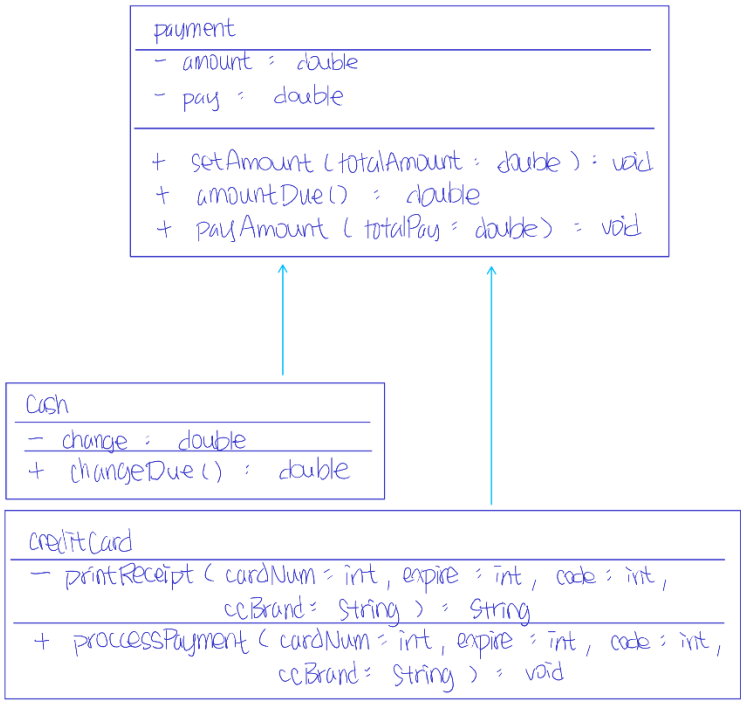
Name: Ivan Leung

Chapter 9

Ex 9.1



Ex 9.7



Ex 9.8

The chid class calls the constructor in the parent class implicitly even if I did not call the constructor of the parent class explicitly. When I call the constructor of the parent class explicitly, there is no difference at all. I believe that the child class is able to call the constructor of the parent class is due to fact that the constructor does not take in any parameters, otherwise there will be compilation errors.

PP 9.1

Source code below:

```
package hw3;
```

```
/*  Java Class: CSCI 145
Author: Ivan Leung
Class: Mon/Wed
Date: May 07 2023
Description:
`
I certify that the code below is my own work.

Exception(s): N/A

*/

public class MonetaryCointTest {

	public static void main(String[] args) {
		final int PENNY = 1;
		final int NICKEL = 5;
		final int DIME = 10;
		final int QUARTER = 25;

		MonetaryCoin mc1 = new MonetaryCoin(PENNY);
		MonetaryCoin mc2 = new MonetaryCoin(NICKEL);
		MonetaryCoin mc3 = new MonetaryCoin(DIME);
		MonetaryCoin mc4 = new MonetaryCoin(QUARTER);

		System.out.println("Flips four coins...");
		mc1.flip();
		mc2.flip();
		mc3.flip();
		mc4.flip();

		System.out.println("Penny lands a " + mc1);
		System.out.println("Nickel lands a " + mc2);
		System.out.println("Dime lands a " + mc3);
		System.out.println("Quarter lands a " + mc1);

		System.out.print("The total value of all coins is ");
		System.out.println((mc1.getValue() + mc2.getValue() +
mc3.getValue() + mc4.getValue()) + " cents");
	}

}

package hw3;
```

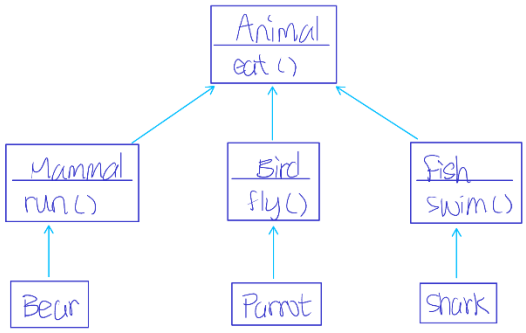
```java
/*  Java Class: CSCI 145
Author: Ivan Leung
Class: Mon/Wed
Date: May 07 2023
Description:
`
I certify that the code below is my own work.

Exception(s): N/A

*/

public class MonetaryCoin extends Coin {
	private int monetaryValue;

	MonetaryCoin(int value) {
		monetaryValue = value;
	}

	public int getValue() {
		return monetaryValue;
	}
}
```

Input/output below:

```
Flips four coins...
Penny lands a Heads
Nickel lands a Tails
Dime lands a Tails
Quarter lands a Heads
The total value of all coins is 41 cents
```

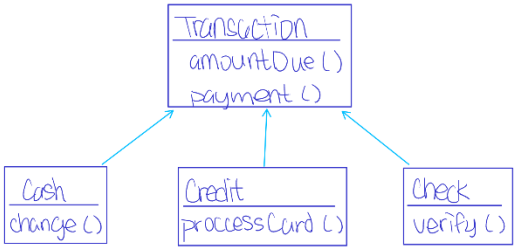
Chapter 10

Ex 10.2



The eat() method in Animal class is an abstract method that in each of the derived classes, Bear, Parrot, and Shark must implement their own version of eat() since in real life, bears are omnivorous, parrots are herbivore, and Sharks are carnivore.

Ex 10.3



Each subclass of Transaction must utilize their own method in order to make payment() method work properly. For example, Credit must use processCard() in payment() to process a credit card payment.

Ex 10.4

If the pay method were not defined as an abstract method, all StaffMember may end up being getting pay the same way including Volunteer which are not supposed to get paid. Also, payRate has to be defined in StaffMember class. However, payRate should not be inherited by Volunteer which are not supposed to get paid.

PP 10.4

Source code below:

```
package hw3;

/*  Java Class: CSCI 145
Modified by: Ivan Leung
Class: Mon/Wed
Date: May 07 2023
Description:
`
I certify that the code below is my own work.

Exception(s): N/A

*/

public class SortingTest {

	public static void main(String[] args) {
		Contact[] friends = new Contact[8];
		friends[0] = new Contact("John", "Smith", "610-555-7384");
		friends[1] = new Contact("Sarah", "Barnes", "215-555-3827");
		friends[2] = new Contact("Mark", "Riley", "733-555-2969");
		friends[3] = new Contact("Laura", "Getz", "663-555-3984");
		friends[4] = new Contact("Larry", "Smith", "464-555-3489");
		friends[5] = new Contact("Frank", "Phelps", "322-555-2284");
		friends[6] = new Contact("Mario", "Guzman", "804-555-9066");
		friends[7] = new Contact("Marsha", "Grant", "243-555-2837");
		Sorting<Contact> sorts = new Sorting<Contact>();

		sorts.selectionSort(friends);
		for (Contact friend : friends)
			System.out.println(friend);
		System.out.println();
		sorts.insertionSort(friends);
		for (Contact friend : friends)
			System.out.println(friend);
	}

}

package hw3;

/*  Java Class: CSCI 145
```

```
Modified by: Ivan Leung
Class: Mon/Wed
Date: May 07 2023
Description:
`
I certify that the code below is my own work.

Exception(s): N/A

*/

//*******************************************************************
//Sorting.java Author: Lewis/Loftus
//
//Demonstrates the selection sort and insertion sort algorithms.
//*******************************************************************

public class Sorting<T> {
	// -----------------------------------------------------------------
	// Sorts the specified array of objects using the selection
	// sort algorithm.
	// -----------------------------------------------------------------
	@SuppressWarnings("unchecked")
	public void selectionSort(Comparable<T>[] list) {
		int max;
		Comparable<T> temp;
		for (int index = 0; index < list.length - 1; index++) {
			max = index;
			for (int scan = index + 1; scan < list.length; scan++)
				if (list[scan].compareTo((T) list[max]) > 0)
					max = scan;
			// Swap the values
			temp = list[max];
			list[max] = list[index];
			list[index] = temp;
		}
	}

	// -----------------------------------------------------------------
	// Sorts the specified array of objects using the insertion
	// sort algorithm.
	// -----------------------------------------------------------------
```

```
	@SuppressWarnings("unchecked")
	public void insertionSort(Comparable<T>[] list) {
		for (int index = 1; index < list.length; index++) {
			Comparable<T> key = list[index];
			int position = index;
			// Shift larger values to the right
			while (position > 0 && key.compareTo((T) list[position
- 1]) > 0) {
				list[position] = list[position - 1];
				position--;
			}
			list[position] = key;
		}
	}
}
```

Input/output below:

```
Smith, Larry    464-555-3489
Smith, John     610-555-7384
Riley, Mark     733-555-2969
Phelps, Frank   322-555-2284
Guzman, Mario   804-555-9066
Grant, Marsha   243-555-2837
Getz, Laura     663-555-3984
Barnes, Sarah   215-555-3827

Smith, Larry    464-555-3489
Smith, John     610-555-7384
Riley, Mark     733-555-2969
Phelps, Frank   322-555-2284
Guzman, Mario   804-555-9066
Grant, Marsha   243-555-2837
Getz, Laura     663-555-3984
Barnes, Sarah   215-555-3827
```

Chapter 11

Ex 11.2

The program will be terminated and all the statement at and after, System.out.println(“Level 1 ending.”), will not be executed, since the exception is not handled in level1().

Ex 11.3

The exception will be handled in level2() therefore, the statements in and after level2() will be executed.

Ex 11.4

a) ArithmeticException indicates that an irregular arithmetic condition has occurred such divide by zero.

b) NullPointerException is thrown when an application tries to access an object's data but the object is null.

c) NumberFormatException indicates when an application tries to convert a string to a numeric type, but the string does not have the correct format.

d) PatternSyntaxException is thrown when there is a syntax error.

PP 11.2

Source code below:

```
package hw3;

/*  Java Class: CSCI 145
Author: Ivan Leung
Class: Mon/Wed
Date: May 07 2023
Description:
`
I certify that the code below is my own work.

Exception(s): N/A

*/

import java.util.Scanner;

public class ExceptionTest {

	public static void main(String[] args) {
		final int MAX_STRING_LENGTH = 20;
		String str;
		Scanner scan = new Scanner(System.in);

		do {
			System.out.println("Enter a string:");
			str = scan.nextLine();
			try {
				if (str.length() >= MAX_STRING_LENGTH)
```

```
                        throw new StringTooLongException("String is
too long!");
                }
                catch (StringTooLongException strExcept) {
                      System.out.println(strExcept.getMessage());
                }
          } while (!str.equalsIgnoreCase("DONE"));
          scan.close();
     }

}
```

```
package hw3;

/*  Java Class: CSCI 145
Author: Ivan Leung
Class: Mon/Wed
Date: May 07 2023
Description:
`
I certify that the code below is my own work.

Exception(s): N/A

*/

@SuppressWarnings("serial")
public class StringTooLongException extends Exception {

     StringTooLongException(String errorMsg) {
          super(errorMsg);
     }

}
```

Input/output below:

```
Enter a string:
hello
Enter a string:
what sup
Enter a string:
this string is really longgggg
```

```
String is too long!
Enter a string:
good bye
Enter a string:
done
```

Chapter 12

Ex 12.2

```
public int power(int x, int y) {
        if (y == 0)
                return 1;
        if (y > 1)
                return x * power(x, y -1);
        else
                return x;
}
```

Ex 12.3

```
public int mul(int i, int j) {
        if (i == 0)
                return 0;
        if (i > 1)
                return j + mul(i – 1, j);
        else
                return j;
}
```

Ex 12.5

```
public int sum(int n, int start) {
        int m = (n + start) / 2;
        if (start > n)
```

```
            return 0;
        if (start == n)
            return n;
        return sum(m, start) + sum(n, m + 1);
}
```

PP 12.1

Source code below:

```
package hw3;

/*  Java Class: CSCI 145
Author: Ivan Leung
Class: Mon/Wed
Date: May 07 2023
Description:
`
I certify that the code below is my own work.

Exception(s): N/A

*/

import java.util.Scanner;

public class PalindromeTester {

	public static void main(String[] args) {
		String str;
		Scanner scan = new Scanner(System.in);

		do {
			System.out.println("Enter a pontential palindrome:");
			str = scan.nextLine();

			System.out.println("\nThat string " +
(isPalindrome(str) ? "IS" : "is NOT") + " a palindrome.\n");

		} while (getChar(scan, "Test another palindrome (y/n)? ") ==
'y');

		scan.close();
```

```
	}

	// Checks a string if it is a palindrome.
	// It ignores any whitespace and punctuation.
	// It is also not case-sensitive.
	public static boolean isPalindrome(String str) {
		str = str.replaceAll("[^a-zA-Z]", ""); // Discards all
whitespace and punctuation.
		str = str.toLowerCase(); // Converts all characters to
lower-case characters.
		return isPalindrome(str, 0, str.length() - 1);
	}

	// Recursive helper method to check for palindrome.
	private static boolean isPalindrome(String str, int start, int
end) {
		if (start >= end)
			return true;
		if (str.charAt(start) != str.charAt(end))
			return false;

		return isPalindrome(str, start + 1, end - 1);
	}

	// Get a char from user input
	private static char getChar(Scanner scan, String prompt) {
		char choice;

		// Input validation
		do {
			System.out.print(prompt);
			choice = scan.nextLine().charAt(0);
			choice = Character.toLowerCase(choice);

			if (Character.compare(choice, 'y') == 0 ||
Character.compare(choice, 'n') == 0)
				return choice;

			System.out.println("Invalid input! Try again!\n\n");
		} while (true);
	}

}
```

Input/output below:

```
Enter a pontential palindrome:
radar

That string IS a palindrome.

Test another palindrome (y/n)? y
Enter a pontential palindrome:
radAr

That string IS a palindrome.

Test another palindrome (y/n)? y
Enter a pontential palindrome:
Iam,Hereh.mai

That string IS a palindrome.

Test another palindrome (y/n)? y
Enter a pontential palindrome:
abcbz

That string is NOT a palindrome.

Test another palindrome (y/n)? n
```