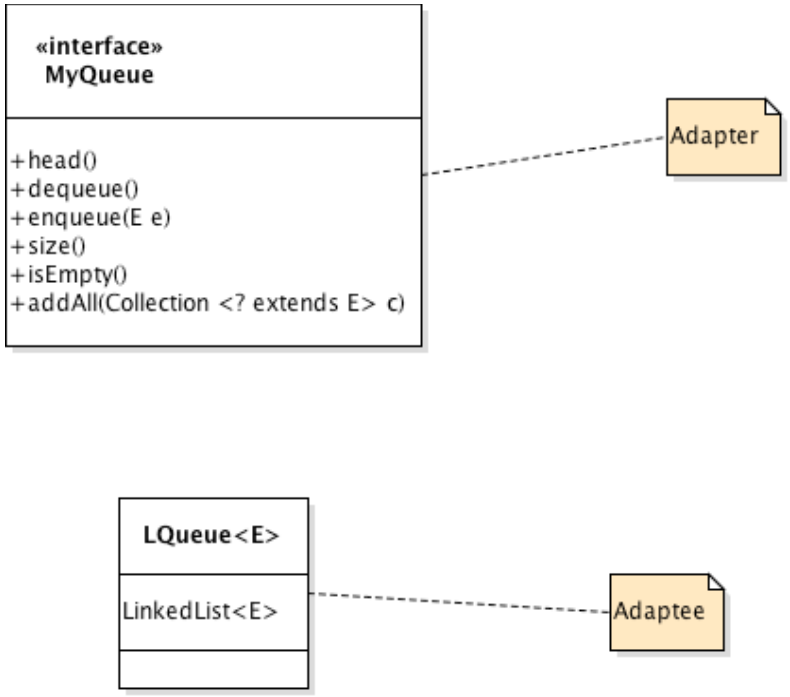

10.1

a.


«interface»
MyQueue
+head()
+dequeue()
+enqueue(E e)
+size()
+isEmpty()
+addAll(Collection <? extends E> c)
Adapter
LQueue<E>
LinkedList<E>
Adaptee

b.

LQueue.java

```
package hw5.queue;

import java.util.Collection;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.NoSuchElementException;

public class LQueue<E> implements MyQueue<E> {

	private LinkedList<E> list;


	public LQueue(){
		this.list = new LinkedList<E>();
	}

	/**
	 * Returns object at head.
	 *
	 * @return the first object in queue
	 * @precondition size() > 0
	 */
	@Override
	public E head() throws NoSuchElementException {
		// TODO Auto-generated method stub
		return this.list.getFirst();
	}

	/**
	 * Returns and Removes object at head.
	 *
	 * @return the object that has been removed from the queue
	 * @precondition size() > 0
	 */
	@Override
	public E dequeue() {
		// TODO Auto-generated method stub
		E item = this.list.removeFirst();
		return item;
	}

	/**
	 * Adds object to end of queue.
	 */
	@Override
	public void Enqueue(E e) {
		// TODO Auto-generated method stub
```

```
			this.list.add(e);
		}

		/**
		 * Returns size of queue.
		 *
		 * @return size of the queue.
		 */
		@Override
		public int size() {
			// TODO Auto-generated method stub
			return this.list.size();
		}

		/**
		 * Checks to see if queue is empty.
		 *
		 * @return true if queue is empty.
		 */
		@Override
		public boolean isEmpty() {
			// TODO Auto-generated method stub
			return this.list.isEmpty();
		}

		/**
		 * Adds all objects in Collection to this queue.
		 */
		@Override
		public void addAll(Collection<? extends E> c) {
			// TODO Auto-generated method stub
			Iterator<? extends E> iter = c.iterator();

			while (iter.hasNext()) {
				E item = (E) iter.next();
				this.Enqueue(item);

			}
		}

}
```

c.

QueueTest.java

```
package hw5.queue;

import java.util.ArrayList;

public class QueueTest {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		LQueue<String> queue = new LQueue<>();

		//Test Enqueue() and Head()
		queue.Enqueue("Enqueue() and Head() worked");
		System.out.println(queue.head());

		//Test Dequeue
		queue.dequeue();
		if (queue.isEmpty()) {
			System.out.println("Dequeue and isEmpty() worked");
		}

		//Test Size
		queue.Enqueue("1");
		queue.Enqueue("2");
		queue.Enqueue("3");
		System.out.print(queue.size());
		System.out.println(" Size() worked");

		//Test addAll
		ArrayList<String> stringArrayList = new ArrayList<>();
		stringArrayList.add("4");
		stringArrayList.add("5");
		stringArrayList.add("6");

		queue.addAll(stringArrayList);

		while (!queue.isEmpty()) {
			System.out.print(queue.dequeue());
		}
		System.out.println(" AddAll() worked");

	}

}
```

10.2

Stdout.java

```java
package hw5.stdout;

public class Stdout {

	private Stdout() {

	}

	public void printline(String s) {
		// print s to System.out
		System.out.println(s);
	}

	public static Stdout getInstance() {
		return instance;
	}

	private static Stdout instance = new Stdout();
}
```

StdoutTest.java

```java
package hw5.stdout;

public class StdoutTest {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Stdout.getInstance().printline("Gregory Prosper");
	}

}
```

10.3

a.

The Command pattern is used to make an object out of *what needs to be done. T*he Strategy pattern, on the other hand, is used to specify *how* something should be done, and plugs into a larger object or method to provide a specific algorithm.

b.

The MouseMotionAdapter class was created for convenience rather than to be compliant with the requirements of an interface.

7.1

a.

Pair.java

```
package hw5.pair;

import java.io.Serializable;

public class Pair<K,V> implements Cloneable,Serializable{

	private K key;
	private V value;

	public Pair(K k, V v){
		this.key = k;
		this.value = v;
	}

	public K k(){
		return this.key;
	}

	public V v(){
		return this.value;
	}

	@Override
	public String toString() {
		return "Pair [key=" + key + ", value=" + value + "]";
	}

	@Override
	public int hashCode() {
		int result = 0;
		result += ((key == null) ? 0 : key.hashCode());
		result += ((value == null) ? 0 : value.hashCode());
		return result;
	}

	@SuppressWarnings("unchecked")
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
```

```
		Pair<K, V> other = (Pair<K, V>) obj;

		if (!this.key.equals(other.key)) {
			return false;
		}
		else if (!this.value.equals(other.value)) {
			return false;
		}
		else{
			return true;
		}
	}

	@SuppressWarnings("unchecked")
	@Override
	public Pair<K,V> clone() {
		// TODO Auto-generated method stub
		try {
			return (Pair<K,V>) super.clone();
		} catch (CloneNotSupportedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
			return null;
		}
	}


}
```

b.

PairTest.java

```
package hw5.pair;

import java.io.*;

public class PairTest {

	@SuppressWarnings("unchecked")
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Pair<String, Integer> pair = new Pair<>("One", 1);
		Pair<String, Integer> pair2 = new Pair<>("One", 2);

		// Test constructer() And toString()
		System.out.println(pair);
```

```
		// Test clone()
		Pair<String, Integer> cloned = pair.clone();
		System.out.println(cloned);

		// Test equals()
		System.out.println(pair.equals(cloned));

		// Test hashCode
		System.out.println(pair.hashCode() == cloned.hashCode());
		System.out.println(pair.hashCode() == pair2.hashCode());

		//Test Serialization
		try {
			FileOutputStream fileOut = new FileOutputStream("/tmp/pair.ser");
			ObjectOutputStream out = new ObjectOutputStream(fileOut);
			out.writeObject(pair);
			out.close();
			fileOut.close();
			System.out.println("Serialized data is saved in /tmp/pair.ser");
		} catch (IOException i) {
			i.printStackTrace();
		}

		Pair<String,Integer> deserializedPair = null;
		try {
			FileInputStream fileIn = new FileInputStream("/tmp/pair.ser");
			ObjectInputStream in = new ObjectInputStream(fileIn);
			deserializedPair = (Pair<String, Integer>) in.readObject();
			in.close();
			fileIn.close();
		} catch (IOException i) {
			i.printStackTrace();
			return;
		} catch (ClassNotFoundException c) {
			System.out.println("Employee class not found");
			c.printStackTrace();
			return;
		}

		System.out.println(deserializedPair);
		System.out.println(pair.equals(deserializedPair));

	}

}
```

7.2

Utils.java

```
package hw5.pair;

import java.util.*;

public class Utils {

	@SuppressWarnings("unchecked")
	public static <K extends Comparable<K>, V> Collection<Pair<K, V>>
sortPairCollection(Collection <Pair<K,V>> col){

		ArrayList<K> array = new ArrayList<>();
		ArrayList<Pair<K,V>> pairArray = new ArrayList<>();

		for (Iterator<Pair<K, V>> iterator = col.iterator(); iterator.hasNext();)
{
			Pair<K, V> pair = (Pair<K, V>) iterator.next();
			array.add(pair.k());
		}
		array.sort(null);

		for (int i = 0; i < array.size(); i++) {
			K key = array.get(i);
			for (Iterator<?> iterator = col.iterator(); iterator.hasNext();) {
				Pair<K, V> pair = (Pair<K, V>) iterator.next();
				if (pair.k() == key) {
					pairArray.add(new Pair<K,V>(key,pair.v()));
				}
			}
		}
		return pairArray;
	}

	public static void main(String[] args) {
		Collection<Pair<String,Integer>> col = new ArrayList<>();
		col.add(new Pair<String,Integer>("Greg",1));
		col.add(new Pair<String,Integer>("Alex",2));
		col.add(new Pair<String,Integer>("Bobby",3));
		col.add(new Pair<String,Integer>("Zack",3));
		col.add(new Pair<String,Integer>("Abel",3));
		col.add(new Pair<String,Integer>("Prince",3));
		col.add(new Pair<String,Integer>("Darrel",3));

		System.out.println(col);
		ArrayList<Pair<String,Integer>> sorted = (ArrayList<Pair<String,
Integer>>) Utils.sortPairCollection(col);
		System.out.println(sorted);
	}
}
```