Assembly Part Code :

Assembly :

```
package warehouse3;

import java.util.ArrayList;

public class Assembly extends Component{

	private ArrayList<Component> components;


	public Assembly(){
		this.components = new ArrayList<Component>();
	}

	public  void add(Component comp){
		this.components.add(comp);
	}

	@Override
	public double cost(){
		double cost = 0.0;
		for(Component comp : components){
			cost+=comp.cost();
		}
		return cost;
	}
}
```

Catalogue Entry :

```
public class CatalogueEntry {

	private String name;
	private long number;
	private double cost;

	public CatalogueEntry(String nme, long number, double cost){
		this.name= nme;
		this.number = number;
		this.cost = cost;
	}

	public String getName(){
		return this.name;
	}

	public long getNumber(){
		return this.number;
	}

	public double getCost(){
		return this.cost;
	}
}
```

Component :

```
package warehouse3;

public abstract class Component {

	public abstract double cost();

}
```

Part :

```
package warehouse3;

public class Part extends Component {

	private CatalogueEntry entry ;

	//constructor
	public Part(CatalogueEntry e) {
		this.entry = e;
	}

	public String getName(){
		return this.entry.getName();
	}

	public long getNumber(){
		return this.entry.getNumber();
	}

	@Override
	public double cost(){
		return this.entry.getCost();
	}
}
```

Service :

```
package warehouse3;

public class Service extends Component {

	private CatalogueEntry entry;

	public Service (CatalogueEntry cat){
		this.entry = cat;
	}

	@Override
	public double cost() {
		return this.entry.getCost();
	}

	//needs getter methods

}
```

Assembly Test :

```
package warehouse3;


import static org.junit.Assert.*;


import org.junit.Test;

public class AssemblyTest {


	@Test
	public void testCost() {

		CatalogueEntry drill = new CatalogueEntry("drill", 28834, 70.00);
		CatalogueEntry lathe = new CatalogueEntry("lathe", 28835, 140.00);
		CatalogueEntry cutter = new CatalogueEntry("tile cutter", 28836, 90.00);
		CatalogueEntry afterSales = new CatalogueEntry("after sales", 2, 10.00);

		Assembly assembly1 = new Assembly();
		assembly1.add(new Part(drill));
		assembly1.add(new Service(afterSales));

		Assembly assembly2 = new Assembly();
		assembly2.add(new Part(drill));
		assembly2.add(new Part(lathe));
		assembly1.add(new Service(afterSales));

		Assembly assembly3 = new Assembly();
		assembly3.add(assembly1);
		assembly3.add(new Part(cutter));
		assembly3.add(new Service(afterSales));

		Assembly assembly4 = new Assembly();
		assembly4.add(assembly2);
		assembly4.add(new Part(lathe));
		assembly3.add(new Service(afterSales));

		Assembly assembly5 = new Assembly();
		assembly5.add(assembly3);
		assembly5.add(new Part(drill));
		assembly5.add(new Part(cutter));
		assembly5.add(new Service(afterSales));
		assembly5.add(assembly4);


		double output = assembly5.cost();
		double expected = 720;

		assertEquals(expected, output, 0.01);
	}
}
```

File System Assignment :
Abstract File :

```
package filesystem;

public abstract class AbstractFile { // abstract class used to tie File and Directory
together into one type

	protected String name; // inherited by subclasses

	public abstract int size(); // abstract method: must be implemented by subclasses
	public abstract int getnumFiles();// abstract method: must be implemented by
subclasses
	public abstract int getnumFolders();// abstract method: must be implemented by
subclasses


	public String getName(){// concrete method: inherited by subclasses
		return name;
	}

}
```

Directory :

```
package filesystem;

import java.util.ArrayList;

public class Directory extends AbstractFile {

	// an ArrayList to hold Files and Directories
	private ArrayList<AbstractFile> files = new ArrayList<AbstractFile>();


	public Directory(String nm){
		this.name = nm; // name is the variable inherited from AbstractFile
	}

	public void add(AbstractFile a) {
		this.files.add(a);

	}

	@Override
	// This method is just like the cost() method in Assembly-Part
	public int size() {
		int size=0;
		for (AbstractFile file: files){
			size+=file.size();
		}
		return size;
	}

	@Override
	public int getnumFiles() {
		int numFiles=0;
		for (AbstractFile file: files){
			numFiles+=file.getnumFiles();
		}
		return numFiles;

	}

	@Override
	public int getnumFolders() {
		int numFolders=0;
		for (AbstractFile file: files){
			if (file instanceof Directory){ //if it is a Directory
				numFolders++; // increase the count
				numFolders+=file.getnumFolders(); // and get the number of
sub-folders it holds
			}
		}
		return numFolders;
	}


}
```

DirectoryTest :

```
package filesystem;

import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class DirectoryTest {

	Directory documents;

	@Before
	public void setUp() throws Exception {

		documents = new Directory("Documents");
		Directory music = new Directory("Music"),
				photos = new Directory("Photos"), dylan = new
Directory("Dylan"), band = new Directory("Band");

		File a = new File("assign1.doc"), b = new File("family.jpg"), c = new
File("tambourine.pm3"),
				d = new File("dixie.mp3"), e = new File("weight.mp3");

		documents.add(a);
		documents.add(music);
		documents.add(photos);
		photos.add(b);
		music.add(dylan);
		music.add(band);
		dylan.add(c);
		band.add(d);
		band.add(e);

	}

	@Test
	public void testSize() {

		int expected = 54;
		int actual = documents.size();
		assertEquals(expected, actual);

	}

	@Test
	public void testGetNumFiles() {
		int expected = 5;
		int actual = documents.getnumFiles();
		assertEquals(expected, actual);

	}

	@Test
	public void testGetNumFolders() {
		int expected = 4;
		int actual =documents.getnumFolders();
		assertEquals(expected, actual);
	}

}
```

File :

```
package filesystem;

public class File extends AbstractFile {

	private int size;

	public File(String nm){
		this.name = nm; // // name is the variable inherited from AbstractFile
		this.size = name.length(); // for demo purposes the size of the file is simply the number of characters in its name
	}

	@Override
	public int size() {
		return this.size;
	}

	@Override
	public int getnumFiles() { // simply returns the value 1
		return 1;
	}

	@Override
	public int getnumFolders() { // we have to implement this (from AbstractFile) - but it doesn't make a lot of sense
		return 0;
	}
}
```

FileSystemDemo :

```
package filesystem;

public class FileSystemDemo {

	public static void main(String[] args) {

		Directory documents = new Directory("Documents"), music = new Directory("Music"),
				photos = new Directory("Photos"), dylan = new Directory("Dylan"), band = new Directory("Band");

		File a = new File("assign1.doc"), b = new File("family.jpg"), c = new File("tambourine.pm3"),
				d = new File("dixie.mp3"), e = new File("weight.mp3");


		documents.add(a);
		documents.add(music);
		documents.add(photos);
		photos.add(b);
		music.add(dylan);
		music.add(band);
		dylan.add(c);
		band.add(d);
		band.add(e);


		// expected results
		int expectedSize = 54;
		int expectedNumFiles = 5;
```

```
		int expectedNumFolders = 4;

		// results returned from the code
		int sizeResult = documents.size();
		int numFilesResult = documents.getnumFiles();
		int numFoldersResult = documents.getnumFolders();


		// testing to see if the expected results match the results returned from
the code
		if(sizeResult==expectedSize){
			System.out.println("size = " + expectedSize + ": correct");
		}else{
			System.out.println("size returned = " + sizeResult + ": incorrect");
		}


		if(numFilesResult==expectedNumFiles){
			System.out.println("numFiles = " + expectedNumFiles + ": correct");
		}else{
			System.out.println("numFiles returned = " + numFilesResult + ":
incorrect");
		}

		if(numFoldersResult==expectedNumFolders){
			System.out.println("NumFolders = " + expectedNumFolders + ":
correct");
		}else{
			System.out.println("numFolders returned = " + numFoldersResult + ":
incorrect");
		}


	}



}
```

Adapter Assignment :
PluginAdapter :

```
package musicplayer;

import java.util.HashMap;

public class PluginAdapter implements Plugin {

	private Plugin player;

	private HashMap <String, Plugin> plugins = new HashMap<String, Plugin>();



	@Override
	public void play(AudioFile audio) throws UnSupportedAudioFormatException {

		this.player = plugins.get(audio.getAudioType());

		if(player==null){
			throw new UnSupportedAudioFormatException("unsupported format: "
					+ audio.getAudioType());
		}

		this.player.play(audio);
	}


	public void registerPlugin(String audioType, Plugin plug) {
		plugins.put(audioType, plug);
	}
}
```

UnSupportedFormatException :

```
package musicplayer;

public class UnSupportedAudioFormatException extends Exception {

	/**
	 *
	 */
	private static final long serialVersionUID = 5600912203505530988L;

	public UnSupportedAudioFormatException(String msg) {
		super(msg);
	}

}
```

WPA Plugin :

```
package musicplayer;

public class WMAPlugin implements Plugin{

	private static String format = "wma";

	@Override
```

```
	public void play(AudioFile audio) throws UnSupportedAudioFormatException{
		if(audio.getAudioType().equals(format)){
			System.out.println("Playing wma file: "+ audio.getFileName());
		}else{
			throw new UnSupportedAudioFormatException("unsupported format: "
					+ audio.getAudioType());
		}
	}
}
```

VLCPlayer:

```
package musicplayer;
//This is a Concrete Music Player

public class VLCPlayer{

	private PluginAdapter adapter = new PluginAdapter();
	private boolean playState = false;


	public void play(AudioFile audio) throws UnSupportedAudioFormatException{

			this.adapter.play(audio);
			this.playState = true;
	}

	public boolean getPlayState(){
		return playState;
	}

	public void stop(){
		this.playState =false;
	}

	public void registerPlugin(String audioType, Plugin plug){
		adapter.registerPlugin(audioType, plug);

	}
}
```

VLCPlayerTest:

```
package musicplayer;

import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;



public class VLCPlayerTest {

	VLCPlayer player;

	@Before
	public void setUp(){
		player = new VLCPlayer();
		player.registerPlugin("ogg", new OGGPlugin());
		player.registerPlugin("mp3", new MP3Plugin());
```

```
		player.registerPlugin("wma", new WMAPlugin());
	}

	// test that the OGG plugin is working with the player
	@Test
	public void testOGG() {
		AudioFile oggFile = new AudioFile("ogg", "C://Music/Amator_Silenti.ogg");
		try{
			player.play(oggFile);
		}catch(UnSupportedAudioFormatException e){
			System.out.println(e.getMessage());
		}
		assertTrue(player.getPlayState()); // assert that the Player is playing
		player.stop();
	}

	// test that the MP3 plugin is working with the player
	@Test
	public void testMP3() {

		AudioFile mp3File = new AudioFile("mp3", "C://Music/Vicissitudes.mp3");
		try{
			player.play(mp3File);
		}catch(UnSupportedAudioFormatException e){
			System.out.println(e.getMessage());
		}

		assertTrue(player.getPlayState()); // assert that the Player is playing
		player.stop();
	}

	// test that the WMA plugin is working with the player
	@Test
	public void testWMA() {
		AudioFile wmaFile = new AudioFile("wma", "C://Music/Lucky_Seven.wma");
		try{
			player.play(wmaFile);

		}catch(UnSupportedAudioFormatException e){
			System.out.println(e.getMessage());
		}
		assertTrue(player.getPlayState()); // assert that the Player is playing
		player.stop();
	}

	// test the correct unsupported audio behaviour
	@Test
	public void testUnsupportedAudio() {
		boolean exceptionThrown = false;

		AudioFile aacFile = new AudioFile("aac", "C://Music/In_Silent_Way.aac");
		try{
			player.play(aacFile);

		}catch(UnSupportedAudioFormatException e){
			System.out.println(e.getMessage());
			exceptionThrown = true; // if the Exception is thrown and caught
		}
		assertTrue(exceptionThrown);
	}

}
```

Plugin :

```
package musicplayer;

public interface Plugin {

	public void play(AudioFile audio)
			throws UnSupportedAudioFormatException;

}
```

OGGPlugin:

```
package musicplayer;

public class OGGPlugin implements Plugin{

	private static String format = "ogg";

	@Override
	   public void play(AudioFile audio) throws UnSupportedAudioFormatException{
	      if(audio.getAudioType().equals(format)){
	      System.out.println("Playing ogg file: "+ audio.getFileName());

	      }else{
			throw new UnSupportedAudioFormatException("unsupported format: "
					+ audio.getAudioType());
	      }
	   }

	}
```

MP3Plugin:

```
package musicplayer;

public class MP3Plugin implements Plugin {

	private static String format = "mp3";

	@Override
	   public void play(AudioFile audio) throws UnSupportedAudioFormatException {

	      if(audio.getAudioType().equals(format)){
	      System.out.println("Playing mp3 file: "+ audio.getFileName());

	      } else{
			throw new UnSupportedAudioFormatException("unsupported format: "
					+ audio.getAudioType());
	      }
	   }
}
```

AudioFile :

```
package musicplayer;

public class AudioFile {

	private String audioType;
	private String fileName;

	public AudioFile (String audioType, String fileName){
		this.audioType = audioType;
		this.fileName = fileName;
```

```
	}

	public String getFileName() {
		return fileName;
	}

	public String getAudioType() {
		return audioType;
	}

}
```

Strategy Grid Assignment:

Avatar :

```
package strategygrid;

public abstract class Avatar {

	private Behaviour behaviour; // behaviour e.g. walk, run or fly
	private Pos currPos; // current position on the grid
	private Grid grid; // reference to the Grid object
	private String name; // Avatar's name

	private Strategy strategy; // strategy determining when to walk run or fly


	public Avatar(String name){
		this.behaviour = new Walk(); //default. Can be changed at runtime
		this.strategy = new Strategy1(this); //default. Can be changed at runtime
		this.name = name;
	}

	public void setBehaviour(Behaviour behave){
		this.behaviour = behave;
	}

	public void setStrategy(Strategy strategy){
		this.strategy = strategy;
	}

	public void setGrid(Grid grid){
		this.grid = grid;
	}

	public void setPos(Pos pos){
		this.currPos = pos;
	}

	public Pos getPos(){
		return this.currPos;
	}

	public boolean move(Pos newPos){
		System.out.print(name + ";");
		if(this.grid.isOccupied(newPos)){
			System.out.println(" Position " + newPos + " is OCCUPIED.");
			return false;
		}

		double dist = Grid.calc(currPos, newPos); // calculate the distance
		System.out.printf(" Distance from " + currPos + " to " + newPos+ " =
%.2f;", dist);
		strategy.setBehaviour(dist); // this method decides the behaviour
		grid.set(newPos, this, behaviour); // update the position
		return true;
	}



	public String getName() {
		return this.name;
	}
}
```

Behaviour:

```
package strategygrid;


public interface Behaviour {

	public void execute(Pos xy);
}
```

Fly:

```
package strategygrid;


public class Fly implements Behaviour {

	@Override
	public void execute(Pos xy) {
		System.out.println(" I will FLY to position " + xy);
	}
}
```

GameEngine:

```
package strategygrid;


//This is just a demo - please customise you own
public class GameEngine {

	// basic demo of the functionality
	public static void main(String[] args) {

		Grid grid = new Grid(10);

		Politician enda  = new Politician("Enda");
		Politician micheal  = new Politician("Michael");
		Politician brendan  = new Politician("Brendan");
		Politician gerry  = new Politician("Gerry");

		//initial positions
		grid.add(enda, new Pos(0,1));
		grid.add(micheal, new Pos(6,2));
		grid.add(brendan, new Pos(4,1));
		grid.add(gerry, new Pos(7,6));

		//for 20 iterations, move them to random locations on the grid
		for (int i = 0; i< 10; i++){
			enda.move(grid.getRandXY());
			micheal.move(grid.getRandXY());
			brendan.move(grid.getRandXY());
			gerry.move(grid.getRandXY());
		}

		System.out.println("Adding New strategy ");
		System.out.println("********************** ");

		new Strategy2(enda);
		new Strategy2(micheal);
		new Strategy2(brendan);
		new Strategy2(gerry);
```

```
		for (int i = 0; i< 10; i++){
			enda.move(grid.getRandXY());
			micheal.move(grid.getRandXY());
			brendan.move(grid.getRandXY());
			gerry.move(grid.getRandXY());
	}


	}
}
```

Grid:

```
package strategygrid;


public class Grid {

	Avatar[][] avatars; // 2-d matrix

	public Grid(int size){
		avatars = new Avatar[size][size];
	}

	public Avatar get(int x, int y){
		return avatars[x][y];
	}

	public void add(Avatar avatar, Pos pos){
		avatars[pos.x][pos.y] = avatar;
		avatar.setGrid(this);
		avatar.setPos(pos);
	}

	public void set(Pos pos, Avatar avatar, Behaviour behave) {

		behave.execute(pos); // execute the behaviour. With a more sophsiticated
approach it will affect how the Avatar moves on the grid
		avatars[avatar.getPos().x][avatar.getPos().y] = null;
		avatars[pos.x][pos.y] = avatar;
		avatar.setPos(pos);
	}

	public boolean isOccupied(Pos pos){
		if (avatars[pos.x][pos.y] != null){
			return true;
		}
		return false;
	}

	public Pos getRandXY(){
		Pos pos = new Pos();
		pos.x = (int) (Math.random() * avatars.length);
		pos.y = (int) (Math.random() * avatars.length);

		return pos;
	}

	public static  double calc(Pos pos1, Pos pos2){

		int temp = (pos1.x - pos2.x) * (pos1.x - pos2.x) + (pos1.y - pos2.y) *
(pos1.y - pos2.y);
		double dist = Math.sqrt(temp);
```

```
            return dist;

      }



}
```

Politician:

```
package strategygrid;

public class Politician extends Avatar {

      public Politician(String name){
            super(name);
      }
}
```

Pos:

```
package strategygrid;


/**
 * @author conorhayes
 * Pos is a simple class that
 * holds (x,y) values
 */
public class Pos {
      public int x,y;

      public Pos(int x, int y){
            this.x = x;
            this.y =y;
      }

      public Pos() {
            this.x = 0;
            this.y =0;
      }

      public String toString(){
            return new String (this.x + "," + this.y);
      }
}
```

Run:

```
package strategygrid;


public class Run implements Behaviour {

      @Override
      public void execute(Pos xy) {
            System.out.println(" I will RUN to position " + xy);
      }
}
```

Strategy:

```
package strategygrid;

public interface Strategy {

      void setBehaviour(double distance);
```

```
}
```

Strategy1:

```
package strategygrid;



/**
 * @author conorhayes
 * We can create new Strategy objects and attach them to the Avatar thus
 * changing how it reacts to the distance variable
 */
public class Strategy1 implements Strategy {

	private Avatar avatar;

	public Strategy1 (Avatar avatar){
		this.avatar = avatar;
	}


	/*
	 * the decide method determines which behaviour to attach to an Avatar
	 * Different strategies will have different decide methods
	 *
	 */
	@Override
	public void setBehaviour(double distance) {
		if(distance < 3){
			avatar.setBehaviour(new Walk());
		} else
			if(distance >= 3 && distance <7){
				avatar.setBehaviour(new Run());
			}
			else{
				avatar.setBehaviour(new Fly());
			}
	}

}
```

Strategy2:

```
package strategygrid;



/**
 * @author conorhayes
 * We can create new Strategy objects and attach them to the Avatar thus
 * changing how it reacts to the distance variable
 */
public class Strategy2 implements Strategy {

	private Avatar avatar;

	public Strategy2 (Avatar avatar){
		this.avatar = avatar;
		this.avatar.setStrategy(this);
	}
```

```
	/*
	 * the decide method determines which behaviour to attach to an Avatar
	 * Different strategies will have different decide methods
	 *
	 */
	@Override
	public void setBehaviour(double distance) {

		double random = Math.random() * distance;

		if(random < 2){
			avatar.setBehaviour(new Walk());
		} else
			if(random >= 2 && random <6){
				avatar.setBehaviour(new Run());
			}
			else{
				avatar.setBehaviour(new Fly());
			}

	}

}
```

Walk:

```
package strategygrid;


public class Walk implements Behaviour {

	@Override
	public void execute(Pos xy) {
		System.out.println(" I will WALK to position " + xy);
	}
}
```

Signatory example code :

Account:

```
package signatory;

import java.util.ArrayList;

public class Account {

	private ArrayList<Signatory> signatories;
	private long number;

	public class Signatory {
		private Customer customer;
		private Account account;

		private Signatory (Customer c, Account a){
			customer = c;
			account = a;
			a.addSignatory(this);
			c.addSignatory(this);
		}

		public Customer getCustomer() {
			return customer;
		}

		public Account getAccount() {
			return account;
		}
	}
	public Account(long n){
		number = n;
		signatories = new ArrayList<Signatory>(3);
	}

	public void makeSignatory(Customer customer){
		new Signatory(customer, this);
	}

	private void addSignatory(Signatory signatory) {
		signatories.add(signatory);
	}

	public ArrayList<Customer> getSignatories() {
		ArrayList<Customer> customers = new
ArrayList<Customer>(signatories.size());
		signatories.forEach(signatory-> customers.add(signatory.getCustomer()));
		return customers;
	}

	public long getNumber() {
		return number;
	}
}
```

Customer:

```
package signatory;

import java.util.ArrayList;

import signatory.Account.Signatory;

public class Customer {

	private ArrayList<Signatory> signatories;
	private String name;

	public Customer(String n){
		name = n;
		signatories = new ArrayList<Signatory>();
	}

	public void addSignatory(Signatory signatory) {
		signatories.add(signatory);
	}

	public String getName() {

		return name;
	}

	public ArrayList<Account> getSignatoryAccounts() {
		ArrayList<Account> accounts = new ArrayList<Account>(signatories.size());
		signatories.forEach(signatory-> accounts.add(signatory.getAccount()));
		return accounts;
	}

}
```

Demo:

```
package signatory;

import java.util.ArrayList;

public class Demo {

	public static void main(String[] args) {

		Account acc1 = new Account(1234);

		Customer customer1 = new Customer("jack");
		acc1.makeSignatory(customer1);

		Customer customer2 = new Customer("maire");
		acc1.makeSignatory(customer2);

		Customer customer3 = new Customer("alicia");
		acc1.makeSignatory(customer3);

		Account acc2  = new Account(4567);
		acc2.makeSignatory(customer1);

		ArrayList<Customer>customers = acc1.getSignatories();

		// note the shortened loop form available in Java 8
		customers.forEach(customer->System.out.println(customer.getName()));

		ArrayList<Account> accounts = customer1.getSignatoryAccounts();
		// note the shortened loop form available in Java 8
		accounts.forEach(account->System.out.println(account.getNumber()));

	}

}
```