CIS 111B

Midterm Exam

Dr. Kendall E. Martin

Submit a complete single document as the final product. Include all code, testing plans, design analysis and commentary that support your solutions.

Question 1 (15 pts) Examine the following code.

**import** java.net.MalformedURLException;

**import** java.net.URL;

**import** java.net.URLConnection;

**import** java.io.IOException;

**import** java.util.Scanner;

**public** **class** Question1 {

@SuppressWarnings({ "resource", "unused" })

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

URLConnection connection = **null**;

**try** {

connection = **new** URL("http://sun.com").openConnection();

} **catch** (IOException e) {

e.printStackTrace();

}

String text = **new** Scanner(connection.getInputStream()).useDelimiter("\\Z").next();

}

}

a)Describe the data type of each part of the code in the last line of the program.

First is a String called text. String is not technically a primitive data type but it acts like one. After which is getInputStream which accepts a data type of string.

b)What is the content of the first 100 characters of text at the end of the program?

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtm

Question 2 (15 pts) In My Programming Lab, complete the assigned problems 20829 and 20839 in chapter 7 section 13, ArrayLists.

20829:

a.set(0,3);

20839:

a.set(0, a.get(a.size() - 1) \* 2);

Question 3 (15 points) **Java API Usage** Use the Java API documentation for Standard Edition 7 to answer the following:

1. List the full prototype for the methods that must be coded if a class implements the java.util interface named Iterator.

public interface Iterator {

public boolean hasNext();

public Object next ();

throws NoSuchElementException

public void remove ();

throws UnsupportedOperationException, IllegalStateException

}

1. How many methods are in the class Track ?

5

1. What package is the class Track located in?

javax.sound.midi

Question 3 (15 pts) **Classes** – Write a class to represent a AlternativeEnergyCar. Select the fields and methods that fit the modeling of an alternative energy car. Make sure to include code for the constructors, set/get methods, a toString() method.

/\*\*

\*

\*/

/\*\*

\* **@author** Alex

\*

\*/

**public** **class** AECar **implements** Comparable<AECar> {

**public** **double** speed;

**public** **double** cost;

**public** String fuelType;

**public** **double** fuelEfficiency;

**public** **double** horsePower;

@Override

**public** String toString() {

**return** "AECar [speed=" + speed + ", cost=" + cost + ", fuelType=" + fuelType + ", fuelEfficiency="

+ fuelEfficiency + ", horsePower=" + horsePower + "]";

}

**public** AECar(**double** speed, **double** cost) {

**super**();

**this**.speed = speed;

**this**.cost = cost;

}

**public** **int** CompareTo(AECar other)

{

**if**(getCost() < other.getCost())

{

**return** 1;

}

**if**(getCost() > other.getCost())

{

**return** -1;

}

**else**

{

**return** 0;

}

}

**public** **double** getSpeed() {

**return** speed;

}

**public** **void** setSpeed(**double** speed) {

**this**.speed = speed;

}

**public** **double** getCost() {

**return** cost;

}

**public** **void** setCost(**double** cost) {

**this**.cost = cost;

}

**public** String getFuelType() {

**return** fuelType;

}

**public** **void** setFuelType(String fuelType) {

**this**.fuelType = fuelType;

}

**public** **double** getFuelEfficiency() {

**return** fuelEfficiency;

}

**public** **void** setFuelEfficiency(**double** fuelEfficiency) {

**this**.fuelEfficiency = fuelEfficiency;

}

**public** **double** getHorsePower() {

**return** horsePower;

}

**public** **void** setHorsePower(**double** horsePower) {

**this**.horsePower = horsePower;

}

/\*\*

\* **@param** args

\*/

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

// **TODO** Auto-generated method stub

}

}

Question 4 (15 pts) **Inheritance** – Create two abstract subclasses of AECar. Next create four additional subclasses. (Note: if you are having difficulty check out some online resources and see what categories of alternative energy cars exist.) Decide which properties should be pushed up into a super abstract class and which belong in the individual subclasses. You do not need to code every method – just place a stub placeholders.

**public** **abstract** **class** Hybrid **extends** AECar {

**double** Emissions;

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

// **TODO** Auto-generated method stub

}

**public** Hybrid() {

}

**public** **void** refuel()

{

}

**public** **void** setEmissions()

{

**this**.Emissions = Emissions;

}

**public** **double** getEmissions()

{

**return** Emissions;

}

}

/\*\*

\*

\*/

/\*\*

\* @author Alex

\*

\*/

public class Prius extends Hybrid {

/\*\*

\*

\*/

double fuelEfficiency;

public double getfuelEfficiency() {

return fuelEfficiency / 2; //Uses half the gas of ordinary Car

}

public void setfuelEfficiency(double fuelEfficiency) {

fuelEfficiency = fuelEfficiency;

}

int Generation;

String EngineType;

public Prius() {

// TODO Auto-generated constructor stub

}

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

}

public int getGeneration() {

return Generation;

}

public void setGeneration(int generation) {

Generation = generation;

}

public String getEngineType() {

return EngineType;

}

public void setEngineType(String engineType) {

EngineType = engineType;

}

}

**public** **class** HondaCivicHybrid **extends** Hybrid {

String EngineType;

**double** fuelEfficiency;

**public** **double** getFuelEfficiency() {

**return** fuelEfficiency;

}

**public** **void** setFuelEfficiency(**double** fuelEfficiency) {

**this**.fuelEfficiency = fuelEfficiency;

}

**public** String getEngineType() {

**return** EngineType;

}

**public** **void** setEngineType(String engineType) {

EngineType = engineType;

}

**public** HondaCivicHybrid() {

// **TODO** Auto-generated constructor stub

}

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

// **TODO** Auto-generated method stub

}

}

/\*\*

\*

\*/

/\*\*

\* @author Alex

\*

\*/

public class NissanLeaf extends ElectricCar {

/\*\*

\*

\*/

public NissanLeaf() {

// TODO Auto-generated constructor stub

}

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

}

String TransmissionType;

public String getTransmissionType() {

return TransmissionType;

}

public void setTransmissionType(String transmissionType) {

TransmissionType = transmissionType;

}

double fuelEfficiency;

public double getFuelEfficiency() {

return fuelEfficiency \* 120; //Converts to Joules

}

public void setFuelEfficiency(double fuelEfficiency) {

this.fuelEfficiency = fuelEfficiency;

}

}

Question 6 (15 pts) **Polymorphism** - Write a driver program that uses a AECar reference variable to point to three different types of classes, showing that inheritance is one mechanism for using polymorphism.

/\*\*

\*

\*/

/\*\*

\* @author Alex

\*

\*/

public class Driver {

/\*\*

\*

\*/

public Driver() {

// TODO Auto-generated constructor stub

}

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

}

AECar.setFuelEfficiency(120);

AECar.getfuelEfficiency();

}

Question 7 (10 pts) **Interface** – Make the AECar class implement the Comparable Java interface. Write a driver program that sorts a list of 4 AECars according to price.

**import** java.util.Arrays;

**public** **class** Driver1 {

**public** Driver1() {

// **TODO** Auto-generated constructor stub

}

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

// **TODO** Auto-generated method stub

AECar other = **new** AECar(200, 2000);

AECar other1 = **new** AECar(220, 5000);

AECar other2 = **new** AECar(270, 5500);

AECar other3 = **new** AECar(280, 5700);

AECar[] aecars = {other,other1,other2,other3};

Arrays.*sort*(aecars);

System.***out***.println("\nHere is a list of aecars sorted by price");

**for**(**int** i = 0; i < aecars.length; i++)

{

System.***out***.println(aecars[i].toString());

}

}

}