# Part A

Q A1.

a).

An abstract class cannot be instantiated

b).

shapes.add(new Square());

c).

Add type cast to shapes.get(0): ((Shape)shapes.get(0)).draw();

Or add the parameter type <Shape> to the arrayList’s instantiation statement:

List<Shape> shapes = new ArrayList<Shape>();

Q A2.

1. List<Person> persons = new ArrayList<Person>();
2. public interface Child{…}

public class Boy extends Person implements Child{…}

public class Girl extends Person implements Child{…}

List<Child> children = new ArrayList<Child>();

children.add(new Boy());

children.add(new Girl());

Q A3.

The **this**(5) refers to another constructor in the same class;

The **this**.getSum() refers to the current object itself

Q A4.

Yes, the ArrayIndexOutOfBoundsException will be caused by the statement a[i+1] = a[i].

Because there are only 5 elements in array a, but the statement above will try to assign the value of the 5th element in a to the non-existent 6th one.

Q A5.

Static methods:

* a.k.a, class methods,can be invoked by the name of the class where it is located;they can also be invoked by the objects instantiated for that class.
* It is illegal to invoke a non-static method from a static method.

Non-static methods:

* Can only be accessed by the name of the objects instantiated for that class where it is located;
* Can invoke a static method;

Q A6.

1. My Exception ! What now?Did we crash?
2. Top-level Exception ! What now? Did we crash?

Q A7.

Public void genError() throws MyException

{

method1();

}

Q A8.

A

Q1

The scope of a superclass reference is limited to the public superclass methods and the overridden methods. To access subclass methods, we need to cast the superclass reference to a subclass reference.

Q2

An abstract class cannot be instantiated;

A subclass must override all abstract methods of its parent, unless it is also declared abstract.

Q3

Q4

An interface can only contain constants and abstract methods.

A class implementing an interface must implement all the abstract methods defined in it.

Q5

1. Derive a class from Thread, override the run() method, then instantiate an object from the derived class as a thread;
2. Create an anonymous Runnable object then pass it to the Thread() constructor to create a Thread object.

public void run(){…}

Q6

1. The class PersonTest was trying to invoke the non-static method print() in class Prize, this is illegal; Rectify: change the method print() in class Prize to static.
2. The method print() in class Prize only prints the current recipient of prizes; Rectify: it should use a for loop to iterate through an array/arrayList which stores the information of recipients and their corresponding prizes;
3. The method getPrizeCount() has never been used; Rectify: either delete this method or change the visibility of variable prizeCount to private.

Q7

The statements in the nested two loops have been executed for 12(i (3)\* j(4)) times.

6 \*s

Q8

ArrayIndexOutOfBoundsException

# Part B

Question B.1(Page 7)

public static void countWords(List<String> list)

{

int count = 0;

for (String str : list)

{

if(str.charAt(0) == 'a' || (str.charAt(0) == 'A'))

count++;

}

System.out.println("Number of a-words: " + count);

System.out.print("Words longer than 5 characters: ");

for (String str : list)

{

if(str.length() > 5)

System.out.print(str + " ");

}

}

Question 9

i.

public static void pop(List list)

{

if(list.size() > 0)

list.remove(list.size() – 1);

}

ii.

public static <T> T(List<T> list)

{

if(list.size() > 0)

{

int lastIndex = list.size() – 1;

T last = list.get(lastIndex);

list.remove(lastIndex);

return Last;

}

Question B.3

JButton hello = new JButton();

hello.setActionCommand(“hello”);

hello.addActionListener(new MyListener());

JButton goodbye = new JButton();

goodbye.setActionCommand(“bye”);

goodbye.addActionListener(new MyActionListener());

Question B.1(Page 9)

public class ReverseArray<T>

{

public <T> ArrayList<T> reverse(T[] o)

{

ArrayList<T> result = new ArrayList<T>();

for(int i = o.length - 1; i >= 0; i--)

{

if(o[i]!= null)

result.add(o[i]);

}

return result;

}

public static void main(String[] args)

{

Integer nums[] = new Integer[3];

nums[0] = 1;

nums[2] = 3;

ReverseArray ra = new ReverseArray();

ArrayList<Integer> result = ra.reverse(nums);

for(Integer i: result)

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

}

}

# Part C

# A Simple Personal Financial Assistant

(a)

public abstract class Item

{

private int day;

private String description;

private double cost;

public Item(int day,String description, double cost)

{

this.day = day;

this.description = description;

this.cost = cost;

}

public double getCost()

{

return cost;

}

public int getDay()

{

return day;

}

public String getDescription()

{

return description;

}

}

public class Bill extends Item

{

private String company;

public Bill(int day, String description, double cost)

{

super(day, description, cost);

}

public Bill(int day, String description, double cost, String company)

{

super(day, description, cost);

company = "unknown";

}

}

public class Stationery extends Item implements TaxDeductable

{

public Stationery(int day, String description, double cost)

{

super(day, description, cost);

}

@Override

public double deduct()

{

return this.getCost() \* 0.25;

}

}

public class Grocery extends Item

{

private static double limit;

public Grocery(int day, String description, double cost)

{

super(day, description, cost);

limit -= cost;

if(limit < 0)

System.out.println("Limit cannot be negative");

}

public static void setLimit(double limit)

{

Grocery.limit = limit;

}

}

public interface TaxDeductable

{

public double deduct();

}

public class Test

{

public static void main(String[] args)

{

Item[] list = new Item[4];

Grocery.setLimit(100.00);

list[0] = new Bill(1, "Phone Bill", 89.99);

list[1] = new Stationery(2, "Printer", 200);

list[2] = new Grocery(3, "Milk and Bread", 20);

list[3] = new Stationery(3, "USB Key", 40);

for(Item i: list)

{

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

}

}

}

Question C.1

public class Employee

{

private int id;

private String name;

private Department department;

private String position;

public Employee(int id, String name, Department department)

{

this.id = id;

this.name = name;

this.department = department;

}

public Employee(int id, String name, Department department, String position)

{

this.id = id;

this.name = name;

this.department = department;

this.position = position;

}

public void setPosition(String posn)

{

position = posn;

}

public void transfer(Department dept)

{

department = dept;

}

public void transfer(Department dept, String posn)

{

department = dept;

position = posn;

}

}

public class Department

{

private String name;

private List<Integer> eList;

public Department(String name, List<Integer> eList)

{

this.name = name;

this.eList = eList;

}

public void addEmployee(int id)

{

this.eList.add(id);

}

public void removeEmployee(int id)

{

this.eList.remove(id);

}

}

# Used Vehicle Dealership System

public abstract class Vehicle

{

private String regNo;

private double purchasePrice;

private double soldPrice;

private int condition;

private boolean sold;

public Vehicle(String regNo, double purchasePrice, Date purchaseDate, int condition)

{

this.regNo = regNo;

this.purchasePrice = purchasePrice;

this.purchaseDate = purchaseDate;

this.condition = condition;

}

public abstract double wantedPrice();

public void sell(double price)

{

try

{

if(this.purchaseDate.numMonthsOld <= 6 && price < wantedPrice() ||

this.purchaseDate.numMonthsOld > 6 && price < wantedPrice() \*

(1 – this.purchaseDate.numMonthsOld \* 0.01) )

throw new PriceTooLowException(“The selling price “ + price + “ is too low !”);

else

{

this.soldPrice = price;

this.sold = true;

}

}

catch(PriceTooLowException ptle)

{

this.soldPrice = null;

this.sold = false;

System.out.println(ptle.getMessage())

}

public class PriceTooLowException extends Exception

{

public PriceTooLowException()

{

super();

}

public PriceTooLowException(String Message)

{

super(message);

}

}

public class Car extends Vehicle

{

private final double BASEPRICE = purchasePrice \* (1 + 0.3);

public Car(String RegNo, double purchasePrice, Date purchaseDate, int condition)

{

super(RegNo, purchasePrice, purchaseDate, condition);

}

@Override

public double wantedPrice()

{

if(this.condition == 1)

return BASEPRICE;

else if(this.condition == 2)

return BASEPRICE \* 0.95;

else if(this.condition == 3)

return BASEPRICE \* 0.90;

}

}

public class Test

{

List<Vehicle> vehicles = new ArrayList<Vehicle>();

vehicles.add(new Car(“OZP413”, 12000, 12:08:2009, 2));

vehicles.add(new MotorCycle(“PQD212”, 4000, 21:07:2009, 3));

vehicles.get(0).sell(18000);

vehicles.get(1).sell(5000);

double totalProfit = 0;

for(Vehicle v : vehicles)

{

totalProfit += v.getSoldPrice() – v.getPurchasePrice();

}

System.out.println(totalProfit);

}

Question D1:

BorderLayout implements LayoutManager2

JLabel jLBank = new JLabel();

jLBank.setText(“THE GOOD BANK”);

JLabel accID = new JLabel();

accID.setText(“Enter account ID: ”);

JLabel accBlance = new JLabel();

accBalance.setText(“Enter its balance: ”)

JTextField id = new JTextfield();

JTextField balance = new JTextField();

JButton addB = new JButton();

addB.setText(“Add”);

JButton saveAllB = new JButton();

saveAllB.setText(“Save All”);

Question D.2

Private HashMap<String, Account> allAccounts = new HashMap<String, Account>();

public boolean saveAll(HashMap<String, Account> allAccounts)

{

boolean saved = false;

String filePath = “allAccounts.txt”;

PrintWriter pw = null;

try

{

pw = new PrintWriter(new FileOutputStream(filePath));

}

catch(FileNotFoundException fnfe)

{

JOptionPane.showDialogMessage(null, fnfe);

}

for(Entry<String, Account> entry: allAccounts.entrySet())

{

String accInfo = “”;

accInfo = “id: “ + entry.getKey() + “;” + “ balance: $” + entry.getValue();

pw.println(accInfo);

}

pw.close();

saved = true;

return saved;

}

Question D.3

addB.setActionCommand(“Add”);

saveAllB.setActionCommand(“Save All”);

public void actionPerformed(ActionEvent e)

{

String cmd = e.getActionCommand();

If(cmd.equals(“Add”)

add(Account ac);

else if(cmd.equals(“Save All”)

saveAll(HashMap<String, Account> allAccounts);

}

public void add(Account ac)

{

ac.setID(id.getText());

ac.setBalance(Double.parseDouble(balance.getText()));

allAccounts.put(ac.getID(), ac);

}

Question D.4

1. Register the event source with a specific listener
2. The user clicks the button with Mouse;
3. The event source instantiates an event object containing information about the event;
4. The event source invokes the event handler method of its registered listener and passes it the event object as a parameter