

מחלקה 1: אובייקט החשבונית (Invoice.java)

```
import java.math.BigDecimal;
import java.util.Date;

public class Invoice {
    private int invoiceId;
    private Integer customerId;
    private Date invoiceDate;
    private Date dueDate;
    private BigDecimal amount;
    private BigDecimal paidAmount;
    private boolean paid;

    // שדה עזר לחישוב ימי איחר (סעיף 3)
    private long daysLate;

    // Constructor
    public Invoice(int invoiceId, Integer customerId, Date invoiceDate, Date dueDate,
BigDecimal amount, BigDecimal paidAmount, boolean paid) {
        this.invoiceId = invoiceId;
        this.customerId = customerId;
        this.invoiceDate = invoiceDate;
        this.dueDate = dueDate;
        this.amount = amount;
        this.paidAmount = paidAmount;
        this.paid = paid;
        this.daysLate = 0;
    }

    // Getters and Setters
    public int getInvoiceId() { return invoiceId; }
```

```

public Integer getCustomerId() { return customerId; }

public Date getDueDate() { return dueDate; }

public BigDecimal getAmount() { return amount; }

public BigDecimal getPaidAmount() { return paidAmount; }

public boolean isPaid() { return paid; }

public long getDaysLate() { return daysLate; }

public void setDaysLate(long daysLate) { this.daysLate = daysLate; }

@Override

public String toString() {

    return "Invoice{" + "id=" + invoiceId + ", customer=" + customerId + ", debt=" +
getDebt() + '}';

}

// פונקציית עזר לחישוב החוב בحسابונית זו

public BigDecimal getDebt() {

    if (amount == null) return BigDecimal.ZERO;

    BigDecimal paid = (paidAmount == null) ? BigDecimal.ZERO : paidAmount;

    return amount.subtract(paid);

}

```

מحلקה 2 (2): הפתרון המלא לפי הסעיפים

Java

```

import java.math.BigDecimal;

import java.math.RoundingMode;

import java.util.*;

public class InvoiceProject {

    public static void main(String[] args) {

```

```

// ייצרת נתונים לדוגמה
List<Invoice> invoices = new ArrayList<>();

Date today = new Date();
תאריך עבר (לפני 10 ימים) // 

Date pastDate = new Date(today.getTime() - (10L * 24 * 60 * 60 * 1000));

invoices.add(new Invoice(1, 101, pastDate, pastDate, new BigDecimal("1000"), new
BigDecimal("200"), false));
invoices.add(new Invoice(2, 102, today, today, new BigDecimal("500"), new
BigDecimal("500"), true));
invoices.add(new Invoice(3, 101, pastDate, pastDate, new BigDecimal("2000"), new
BigDecimal("0"), false));
invoices.add(new Invoice(4, 103, today, today, new BigDecimal("300"), new
BigDecimal("100"), false));

---דרישה 1: חישובים מתקדמים---
System.out.println("---");
calculateAdvancedMath(invoices);

---דרישה 2: מקרי קצה בחילוק---
System.out.println("\n---");
demonstrateDivisionEdgeCases();

---דרישה 3: עבודה עם תאריכים---
System.out.println("\n---");
handleDates(invoices);

---דרישה 4: מין וצבירה---
System.out.println("\n---");
Map<Integer, BigDecimal> customerDebts = mapCustomerDebts(invoices);
sortAndPrintCustomerDebts(customerDebts);

}

-דרישה 1: חישובים ב--- //
public static void calculateAdvancedMath(List<Invoice> invoices) {
    BigDecimal totalOpenDebt = BigDecimal.ZERO;
}

```

```

BigDecimal totalAmountAll = BigDecimal.ZERO;
List<Invoice> openDebtInvoices = new ArrayList<>();

for (Invoice inv : invoices) {
    BigDecimal debt = inv.getDebt();
    totalAmountAll = totalAmountAll.add(inv.getAmount());
}

// סכום חוב פתוח
totalOpenDebt = totalOpenDebt.add(debt);

// רשימת חשבוניות עם חוב גדול מ-0
if (debt.compareTo(BigDecimal.ZERO) > 0) {
    openDebtInvoices.add(inv);
}

}

System.out.println("סה"כ חוב פתוח לכל החשבוניות;" + totalOpenDebt);
System.out.println("מספר חשבוניות עם חוב;(" + openDebtInvoices.size() + ")");

// חישוב אחוז החוב מתוך הסכום הכללי
if (totalAmountAll.compareTo(BigDecimal.ZERO) != 0) {
    BigDecimal percentage = totalOpenDebt.divide(totalAmountAll, 2,
RoundingMode.HALF_UP)
        .multiply(new BigDecimal("100"));
    System.out.println(" אחוז החוב מתוך הסכום הכללי;" + percentage + "%");
}

}

// --- דרישת 2: דוגמת קוד לחילוק ב-0 ---
public static void demonstrateDivisionEdgeCases() {
    BigDecimal num = new BigDecimal("100");

```

```

BigDecimal zero = BigDecimal.ZERO;

// BigDecimal
try {
    num.divide(zero);
} catch (ArithmaticException e) {
    System.out.println("BigDecimal divide by 0 error: " + e.getMessage());
}

// Double
double dNum = 100.0;
double dZero = 0.0;
Infinity  יופיע System.out.println("Double divide by 0 result: " + (dNum / dZero)); //

}

--- // דרישת 3: עבודה עם תאריכים---

} public static void handleDates(List<Invoice> invoices)
;()Date today = new Date

} for (Invoice inv : invoices)

// טיפול בטעות ב-null
BigDecimal paid = (inv.getPaidAmount() == null)
BigDecimal.ZERO ?
;()inv.getPaidAmount :

;boolean isOverdue = inv.getDueDate().before(today)
;boolean isNotFullyPaid = paid.compareTo(inv.getAmount()) < 0

} if (isOverdue && isNotFullyPaid)
;()long diffInMillies = today.getTime() - inv.getDueDate().getTime

```

```

;(24 * 60 * 60 * 1000) / long diffInDays = diffInMillies

;inv.setDaysLate(diffInDays)

)System.out.println
+ ()inv.getInvoiceId + "חובונית "
" באיחור של " + diffInDays + " ימים."
;(
{
{
{
--- // דרישת 4: מין וצבריה---

public static Map<Integer, BigDecimal> mapCustomerDebts(List<Invoice> invoices) {
    Map<Integer, BigDecimal> debtsMap = new HashMap<>();

    for (Invoice inv : invoices) {
        BigDecimal currentDebt = inv.getDebt();
        if (currentDebt.compareTo(BigDecimal.ZERO) > 0) {
            Integer custId = inv.getCustomerId();
            debtsMap.put(custId, debtsMap.getOrDefault(custId,
                BigDecimal.ZERO).add(currentDebt));
        }
    }
    return debtsMap;
}

public static void sortAndPrintCustomerDebts(Map<Integer, BigDecimal> map) {
    // המרת המפה לרשימה כדי לMIN
    List<Map.Entry<Integer, BigDecimal>> list = new ArrayList<>(map.entrySet());
    // מין לפי הערך (Value) מהגבוה לנמוך
    list.sort((entry1, entry2) -> entry2.getValue().compareTo(entry1.getValue()));
}

```

```
    : חובות לקוחות (מהגבהה לנור);  
    System.out.println("  
    for (Map.Entry<Integer, BigDecimal> entry : list) {  
        לקוחות | :" + entry.getValue() + " : " + entry.getKey() + "  
        System.out.println("}  
    }  
}
```

חלק עיוני: תשובות לשאלות

סעיף 2: טיפול במרקורי קצה בחילוק

- **חלוקת ב-0 עם BigDecimal:** הפעולה תזרוק שגיאה מסוג ArithmeticException (ז裏יקת חריגה), כיוון שמתמטית הפעולה אינה מוגדרת והמחלקה בנייה לדיק מוחלט. התוכנית תעצור אם לא נעתוף זאת ב-try-catch.
 - **חלוקת ב-0 עם float או double:** הפעולה לא תזרוק שגיאה. במקום זאת, נקבל תוצאה מיוחדת בשם NaN או Infinity - אם המחלק שלילי). הסיבה לכך היא הדרך שבה המעבד מטפל בנקודות צפה (Floating Point standard IEEE 754).

סעיף 5: טיפוסים פרימיטיביים מול עטופים

- אם השדה **customerId** היה `null` וקיים טיפוס פרימיטיבי (`int`) לא יכול להציג ערך null. ננסה להכניס `null` למשתנה, `int`ותהיה שגיאת קומפליציה (Unboxing of null) אם זה קורה בזמן המרת `Integer`-int, תריה שגיאות זמן ריצה (`NullPointerException`). לכן, השימוש ב-`Integer`-עדיף כיון שהוא מאפשרלי'יצג מצב שבו "אין לך" (עריך ריק).

2. השפעה על שימוש ב-**Collections**- מבני נתונים ב Java- כמו List, Map או Setעובדים רק עם אובייקטים Generics.

- או אפשר להגדיר `Map<int, BigDecimal>`

- : Map<Integer, BigDecimal> מעתפת. השם כוחה להשתמש

- 3. מה יקרה אם paidAmount הוא null בחישוב: השורה :**

- התוצאה: תהיה שגיאת זמן ריצה** (Runtime Exception) מסוג paidAmount.compareTo(amount) < 0

- הסביר:** א' אפשר לקרוא למתודה compareTo (השווה) על אובייקט שהוא null.

- תיקון הקוד:** יש לבדוק האם הערך הוא `null` לפני ההשוואה, או לתת לו ערך ברירת מחדל של 0.

דוגמת תיקון:

Java

//**אפקיה א'**: בדיקה מקדים

```
if (paidAmount != null && paidAmount.compareTo(amount) < 0) { ... }

//אפשריה ב': שימוש בערך ברירת מחדל (מולץ)
BigDecimal safePaid = (paidAmount == null) ? BigDecimal.ZERO : paidAmount;
if (safePaid.compareTo(amount) < 0) { ... }
```