

מחלקה 1) Invoice.java: אובייקט החשבונית)

```
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 איחור (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 בחשבונית זו
public BigDecimal getDebt() {

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

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

    return amount.subtract(paid);

}
}

```

מחלקה 2) InvoiceProject.java: 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) {

```

```

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

--- // דרישה 1: חישובים ב-BigDecimal ---
    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 הכולל
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 הכולל: " + percentage + "%");
}

--- // דרישה 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 (ArithmeticException e) {
```

```
    System.out.println("BigDecimal divide by 0 error: " + e.getMessage());
```

```
}
```

```
// Double
```

```
double dNum = 100.0;
```

```
double dZero = 0.0;
```

```
Infinity ידפיס System.out.println("Double divide by 0 result: " + (dNum / dZero)); //
```

```
}
```

```
--- דרישה 3: עבודה עם תאריכים ---
```

```
} public static void handleDates(List<Invoice> invoices)
```

```
;())Date today = new Date
```

```
} for (Invoice inv : invoices)
```

```
    null טיפול בטוח ב-//
```

```
    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 + "חשבונית"
```

```
" באיחור של " + diffInDays + " ימים."
```

```
;(
```

```
{
```

```
{
```

```
{
```

```
--- דרישה 4: מיון וצבירה ---
```

```
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 למיין
```

```
List<Map.Entry<Integer, BigDecimal>> list = new ArrayList<>(map.entrySet());
```

```
// מיון לפי הערך (Value) מהגבוה לנמוך
```

```
list.sort((entry1, entry2) -> entry2.getValue().compareTo(entry1.getValue()));
```

```

System.out.println("
חובות לקוחות (מהגבוה לנמוך);");
for (Map.Entry<Integer, BigDecimal> entry : list) {
    System.out.println("
לקוח | " + entry.getKey() + " :חוב: " + entry.getValue());
}
}
}
}

```

חלק עיוני: תשובות לשאלות

סעיף 2: טיפול במקרי קצה בחילוק

- **חילוק ב-0 עם BigDecimal:** הפעולה תזרוק שגיאה מסוג `ArithmeticException` (זריקת חריגה), כיוון שמתמטית הפעולה אינה מוגדרת והמחלקה בנויה לדיוק מוחלט. התוכנית תעצור אם לא נעטוף זאת ב-try-catch.
- **חילוק ב-0 עם double או float:** הפעולה לא תזרוק שגיאה. במקום זאת, נקבל תוצאה מיוחדת בשם `Infinity` או `-Infinity` (אם המחלק שלילי). הסיבה לכך היא הדרך שבה המעבד מטפל בנקודה צפה. (Floating Point standard IEEE 754)

סעיף 5: טיפוסים פרימיטיביים מול עטופים

1. **אם השדה `customerId` היה `int` וקיבלנו `null`:** טיפוס פרימיטיבי (`int`) לא יכול להחזיק ערך `null`. ננסה להכניס `null` למשתנה `int`, תהיה שגיאת קומפילציה. (Unboxing of null) אם זה קורה בזמן המרה מ-`Integer` ל-`int`, תהיה שגיאת זמן ריצה. (`NullPointerException`). לכן, השימוש ב-`Integer` עדיף כיוון שהוא מאפשר לייצג מצב שבו "אין לקוח" (ערך ריק).

2. **השפעה על שימוש ב-`Collections`:** מבני נתונים ב-Java כמו `Map`, `List` או `Set` עובדים רק עם אובייקטים. (Generics)

○ אי אפשר להגדיר `Map<int, BigDecimal>`.

○ חובה להשתמש במעטפת `Map<Integer, BigDecimal>`.

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

```
paidAmount.compareTo(amount) < 0
```

○ **התוצאה:** תהיה שגיאת זמן ריצה (`Runtime Exception`) מסוג `NullPointerException`.

○ **הסבר:** אי אפשר לקרוא למתודה (`compareTo`) 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

//אופציה א': בדיקה מקדימה

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

//אופציה ב': שימוש בערך ברירת מחדל (מומלץ)

```
BigDecimal safePaid = (paidAmount == null) ? BigDecimal.ZERO : paidAmount;
```

```
if (safePaid.compareTo(amount) < 0) { ... }
```