Q1:

Part (a):

```
public static int arraysum(int[] arr) {
    int sum = 0;
    for (int elem : arr) {
        sum += elem;
    }
    return sum;
}
```

Part (b):

```
public static int[] rowsums(int[][] arr2D) {
    int[] sums = new int[arr2D.length];
    int rowNum = 0;
    for (int[] row : arr2D) {
        sums[rowNum] = arraysum(row);
        rowNum++;
    }
    return sums;
}
```

Part (c):

```
public static boolean isDiverse(int[][] arr2D) {
	int[] sums = rowsums(arr2D);
	for (int i = 0; i < sums.length; i++) {
		for (int j = i + 1; j < sums.length; j++) {
			if (sums[i] == sums[j]) {
				return false;
			}
		}
	}
	return true;
}
```

Q2:

```
public class HiddenWord {
	private String word;

	public HiddenWord(String hWord) {
		word = hWord;
	}

	public String getHint(String guess) {
```

```
        String hint = "";
        for (int i = 0; i < guess.length(); i++) {
            if (guess.substring(i, i + 1).equals(word.substring(i, i
+ 1))) {
                hint += guess.substring(i, i + 1);
            } else if (word.indexOf(guess.substring(i, i + 1)) != -1)
{
                hint += "+";
            } else {
                hint += "*";
            }
        }
        return hint;
    }
}
```

Q3:

Part (a):

```
public int getValueAt(int row, int col) {
    for (SparseArrayEntry e : entries) {
        if (e.getRow() == row && e.getCol() == col) {
            return e.getValue();
        }
    }
    return 0;
}
```

Part (b):

```
public void removeColumn(int col) {
    int i = 0;
    while (i < entries.size()) {
        SparseArrayEntry e = entries.get(i);
        if (e.getCol() == col) {
            entries.remove(i);
        } else if (e.getCol() > col) {
            entries.set(i, new SparseArrayEntry(e.getRow(),
                e.getCol() - 1,
                e.getValue()));
```

```
                i++;
            } else {
                i++;
            }
        }
        numCols--;
}
```

Q4:

Part (a):

javaCopy code

```
public interface NumberGroup {
        boolean contains(int num);
}
```

Part (b):

```
public class Range implements NumberGroup {
        private int min;
        private int max;

        public Range(int min, int max) {
            this.min = min;
            this.max = max;
        }
```

```
    public boolean contains(int num) {
        return num >= min && num <= max;
    }
}
```

Part (c):

```
public boolean contains(int num) {
    for (NumberGroup group : grouplist) {
        if (group.contains(num)) {
            return true;
        }
    }
    return false;
}
```