BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
FIRST SEMESTER, 2018-2019
MID SEMESTER EXAM - SOLUTIONS
OBJECT ORIENTED PROGRAMMING (CS F213)

DATE: 12th October 2018 **MAX MARKS: 60** **WEIGHTAGE: 30%** **TIME: 90 Min**

Q1. a)

```
int mid;
if (low>high){
        System.out.println("No such book found");
        return -1;
}
        mid = low+(high-low)/2;
```
1.5M

```
    if( list.get(mid).compareTo(key) > 0 )
        return binarySearch(list, key, mid+1, high);
     else if(list.get(mid).compareTo(key) < 0 )
        return binarySearch( list, key, low, mid-1 );
```
3M

```
     else
        return mid;
```
0.5M

Q1. b)

```
int n= list.size();
```
0.5M

```
switch(code) {
case 0: ArrayList<Integer> al1 = new ArrayList<Integer>();
                        for(int i = 0;i<n;i++)
                         al1.add(i, list.get(i).bookID);
                        return al1;
case 1: ArrayList<String> al2 = new ArrayList<String>();
                for(int i = 0;i<n;i++)
                        al2.add(i, list.get(i).bookName);
                return al2;
        }
```
2M
2M

```
return null;
```
0.5M

Q1. c)

```
al.add(new Library("Java",17888,"A2"));
al.add(new Library("C++",19532,"F1"));
al.add(new Library("IoT",242537,"B6"));
al.add(new Library("DIP",347888,"E8"));
```
2M

Q1. d)

```
Collections.sort(al,new Comparator<Library>(){
```
2M

```
public int compare(Library l1, Library l2) {
        if (l1.bookID==l2.bookID)
                return 0;
        else if(l1.bookID<l2.bookID)
                return 1;
        else
                return -1;
        }
});
```
2M

Q1. e) Collections.*sort*(al,**new** Comparator<Library>(){ 2M

```
public int compare(Library l1, Library l2) {
        if(l1.bookName.equals(l2.bookName))
                return 0;
                else if(l1.bookName.compareTo(l2.bookName)<0)
                return 1;
                else
                return -1;
        }
});
```

2M

Q3. leaving testIt 1M
5

leaving testIt 1M
10

leaving testIt 1M
Index exception

Goodbye World 1M

Q4. this.y in line no. 8 should be replaced with y 1M
Output is:
value of x and y = 1, 55, 0 2M

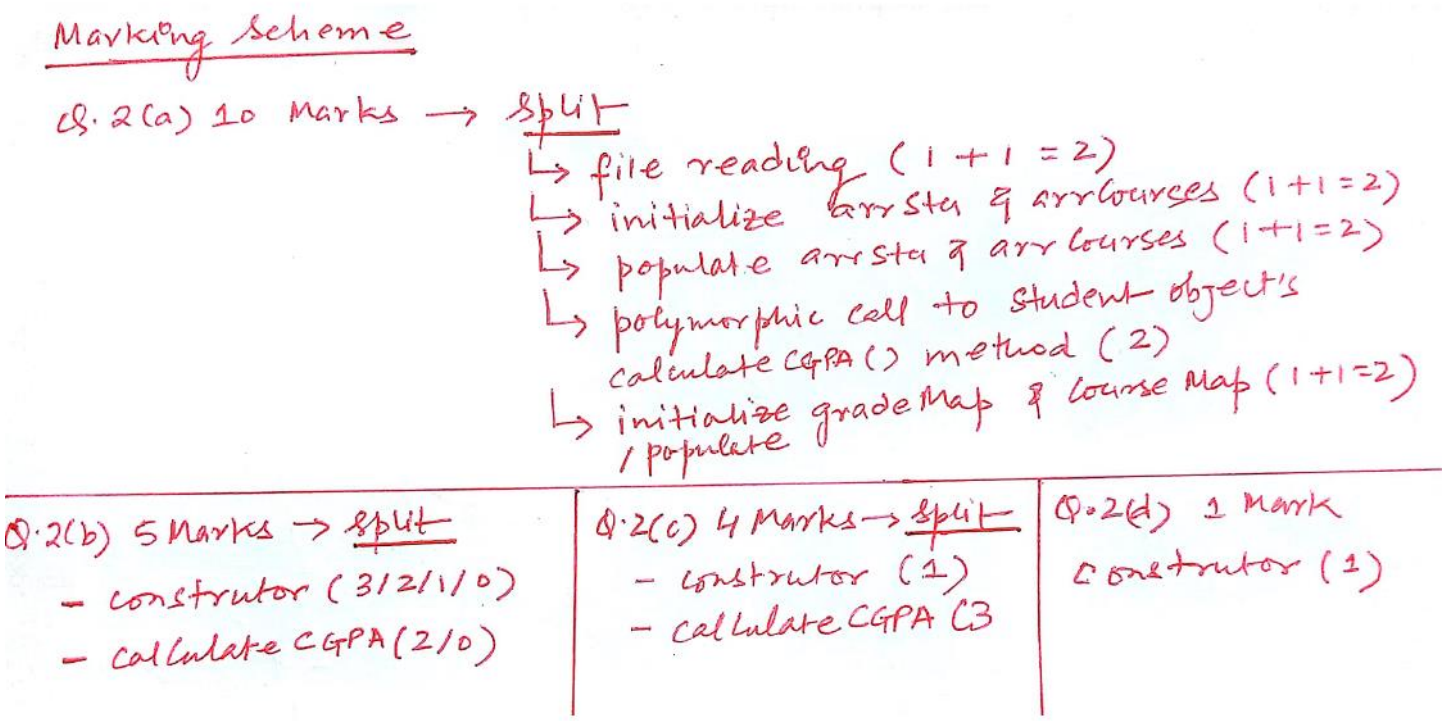
Q5. (g) B is a supertype of C 1M
(j) B is a subtype of D 1M
(l) C is a subtype of D 1M

Q6.

- Unchecked exceptions generally indicate programming errors. They could appear anywhere in the code and documenting them would pointlessly clutter up interfaces.
- Checked exceptions represent unusual or unexpected events such as being unable to open a file or a dropped network connection. While they are not necessarily expected to occur, they are things a client might want to handle. It is appropriate to document them as part of a method's interface if they might be thrown by the method, or require that the method catch them if they should not be visible outside.

Q7. (a) Parent 2M
(b) short version: 10 1M
long version: 20 1M
short version: 20 1M
(c) Length: 14 1M
Capacity: 30 1M

Q2.




```
package test;

public class Course {

	private String code;
	private int units;

	public Course(String code, int units) {
		super();
		this.code = code;
		this.units = units;
	}

	public String getCode() {
		return code;
	}

	public int getUnits() {
		return units;
	}

}
```

```
package test;


public abstract class Student {

	protected double cgpa;
```

```
	protected String id;
	protected int no_of_courses;
	protected String[] courses;
	protected String[] grades;

	public Student(String[] record) {

		this.no_of_courses = Integer.parseInt(record[1]);

		this.id = record[2];

		this.courses = new String[no_of_courses];
		this.grades = new String[no_of_courses];

		int i=3;

		for(int j=0; j<no_of_courses; j++, i++)
			courses[j] = record[i];

		int x = i;

		for(int j=0; i<(x+no_of_courses); j++, i++)
			grades[j] = record[i];

	}

	public double calculateCGPA() {

		int totalUnits = 0;
		double gradeWeight = 0;

		for(int i=0; i<no_of_courses; i++) {
			Integer mulFactor = GradingSystem.getGrademap().get(grades[i]);
			Course course = GradingSystem.getCoursemap().get(courses[i]);
			int units = course.getUnits();
			totalUnits += units;
			gradeWeight += units * mulFactor;
			cgpa = gradeWeight/totalUnits;
		}

		return cgpa;
	}
}



package test;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;

public class GradingSystem {

	private static Student[] arrStu = null;
	private static Course[] arrCourses = null;

	private static final HashMap<String, Integer> gradeMap = new HashMap<String, Integer>();
	private static final HashMap<String, Course> courseMap = new HashMap<String, Course>();
```

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

       String str;
       int line_no = 0;

       BufferedReader br1 = null;
    BufferedReader br2 = null;

    try {

       br1 = new BufferedReader(new FileReader("F:\\student.txt"));
       br2 = new BufferedReader(new FileReader("F:\\courses.txt"));

        //initialize student array
       while ((str = br1.readLine()) != null) { line_no++; }
        arrStu = new Student[line_no];

        line_no = 0; //reset line_no

        //initialize courses array
        while ((str = br2.readLine()) != null) { line_no++; }
        arrCourses = new Course[line_no];

        line_no = 0; //reset line_no

        br1 = new BufferedReader(new FileReader("F:\\student.txt")); //reinitialize br1
       br2 = new BufferedReader(new FileReader("F:\\courses.txt")); //reinitialize br2

        //initialize elements of student array
        while ((str = br1.readLine()) != null) {

              String[] st = str.split("~|#");

              if(st[0].equalsIgnoreCase("ug")){
                     arrStu[line_no] = new UG(st);
              } else {
                     arrStu[line_no] = new PG(st);
              }

              line_no++;
        }

        line_no = 0; //reset line_no

        //initialize elements of courses array      - ask them to use string tokenizer here
        while ((str = br2.readLine()) != null) {

              String[] st = str.split("~");

              arrCourses[line_no] = new Course(st[0], Integer.parseInt(st[1]));

              line_no++;
        }

        initializeGradeMap();
        initializeCourseMap();
```

```
            } catch (IOException e) {
                    e.printStackTrace();
            }

            //exhibit polymorphism and find cgpa of all students
            for(Student s : arrStu) {
               s.calculateCGPA();
               System.out.println(s.toString());
            }

      }

      private static void initializeGradeMap(){
            gradeMap.put("A", 10);
            gradeMap.put("A-", 9);
            gradeMap.put("B", 8);
            gradeMap.put("B-", 7);
            gradeMap.put("C", 6);
            gradeMap.put("C-", 5);
            gradeMap.put("D", 4);
            gradeMap.put("E", 2);
      }

      private static void initializeCourseMap(){
            for(int i=0; i<arrCourses.length; i++){
                  courseMap.put(arrCourses[i].getCode(), arrCourses[i]);
            }
      }

      public static HashMap<String, Integer> getGrademap() {
            return gradeMap;
      }

      public static HashMap<String, Course> getCoursemap() {
            return courseMap;
      }
}



package test;

public class PG extends Student {

      private boolean thesis;

      public PG(String[] record) {
            super(record);
            if("Thesis".equalsIgnoreCase(record[3]))
                  this.thesis = true;

      }

      public double calculateCGPA() {

            if(false != thesis){

                  Integer mulFactor = GradingSystem.getGrademap().get(grades[0]);
                  Course course = GradingSystem.getCoursemap().get("Thesis");
```

```
                int units = course.getUnits();

                if(mulFactor >= 8)
                        cgpa = (units * mulFactor)/units + 0.1;
                else
                        cgpa = (units * mulFactor)/units;
        }

        else {
                super.calculateCGPA();
        }

        return cgpa;
    }

    @Override
    public String toString() {
        return "PG [ id = " + id + " cgpa = " + cgpa + "]";
    }


}


package test;

public class UG extends Student {

    public UG(String[] record) {
        super(record);
    }

    public String toString() {
        return "UG [ id = " + id + " cgpa = " + cgpa + "]";
    }

}
```