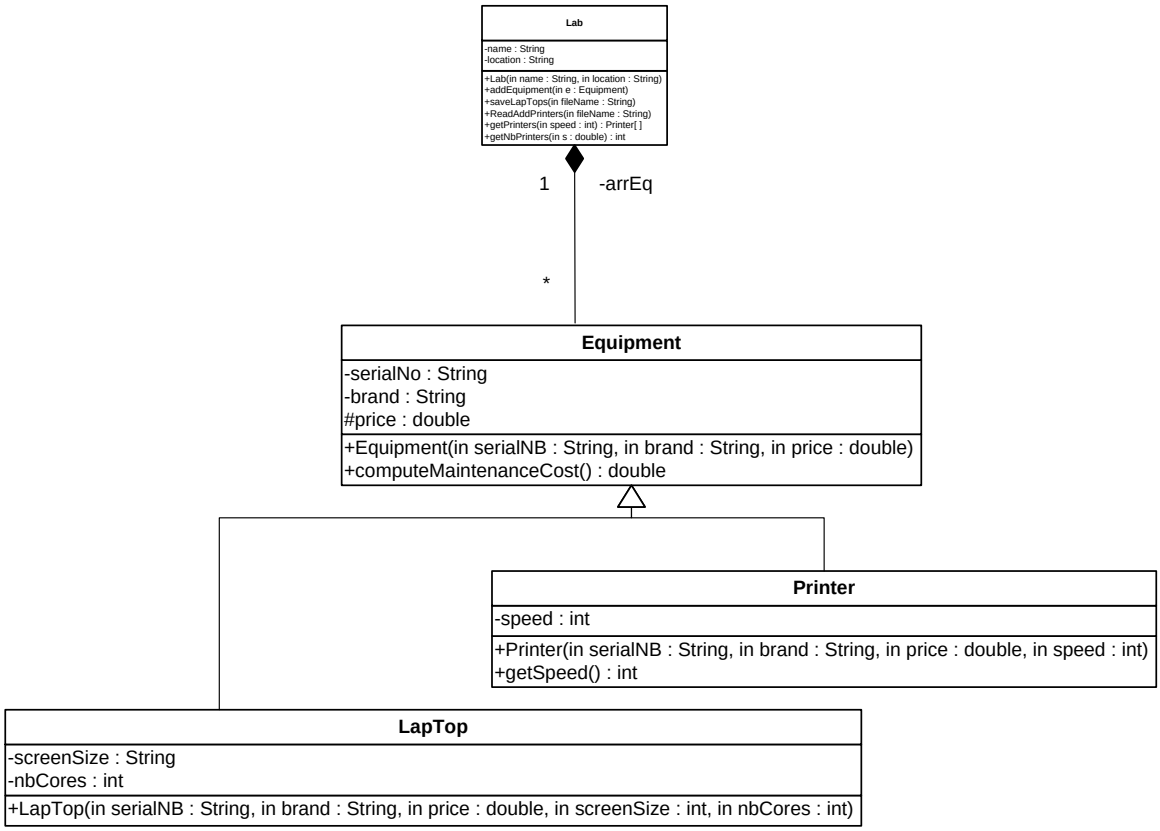
King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC113 – Computer Programming II – Final Exam – Spring 2015

Exercise 1:

```
public class PersonalComputer {
	private String brand;
	protected int capacity;

	public PersonalComputer() {  brand = "Unknown";		capacity = 10;
		System.out.println("PersonalComputer defalut constructor");
	}
	public PersonalComputer(String brand, int capacity) {
		this.brand = brand;		this.capacity = capacity;
		System.out.println("PersonalComputer object created");
	}

	public void display() throws Exception {
		if(brand.equals("Unknown"))
			throw new Exception("brand is Unknown");
		System.out.println("Brand: " + brand + " Capacity: " + capacity);
	}

	public int cost() throws Exception {
		if (capacity < 20)
			throw new Exception("Capacity Less Than 20 GB");
		return 200;
	}
}

public class Desktop extends PersonalComputer {
	protected int nbPeripherals;

	public Desktop() {
		System.out.println("Desktop defalut constructor");
		nbPeripherals = 3;
	}
	public Desktop(int nbPeripherals) {
		System.out.println("Desktop 1st constructor");
		this.nbPeripherals = nbPeripherals;
	}
		public Desktop(String brand, int capacity, int nbPeripherals) {
		super(brand, capacity);
		System.out.println("Desktop 2nd constructor");
		this.nbPeripherals = nbPeripherals;
	}
	public void display() throws Exception {
		super.display();
		System.out.println("Desktop with "+nbPeripherals+" peripherals");
		System.out.println("With cost = "+cost());
	}
	public int cost() throws Exception {
		return (super.cost() + capacity * nbPeripherals);
	}
}
```

```
public class TestPC {
	public static void main(String[] args) {
		int i;
		PersonalComputer[] pclist = new PersonalComputer[3];

		pclist[0] = new Desktop("Dell", 200, 3);
		pclist[1] = new Desktop(5);
		pclist[2] = new Desktop("Toshiba",15,5);

		for (i = 0; i < 4; i++) {
			System.out.println("Iteration " + (i + 1));
			try {
				pclist[i].display();
			} catch (ArrayIndexOutOfBoundsException e) {
				System.out.println("Exception in Iteration " + (i + 1) +
							" Index out of bound");
			}
			catch(Exception e)
			{
				System.out.println("Exception in Iteration " + (i + 1) +
							" " + e.getMessage());
			}
		}// end for
	}// end main
}
```

Exercise2:



Equipment class***:***

- Attributes:
  - ***serialNo***: the serial number of the equipment.
  - ***brand:*** the brand of the equipment.
  - ***price:*** the price of the equipment.
- Methods:
  - ***Equipment (serialNB: String, brand: String, price: double)***: constructor
  - ***computeMaintenanceCost()*:** this method returns the annual cost of the maintenance of the equipment. This cost is computed as following:
    - ***For Laptops:*** the cost = 1 % of price + nbCores * 200
    - ***For Printers:*** the cost = 2 % of price + speed * 100.

LapTop class

- Attributes:
  - ***screenSize:*** the size of the screen of the laptop.

- ***nbCores***: the number of core processors of the laptop.

- Methods:
  - ***LapTop (serialNB:** String**, brand:** String**, price:** double**, screenSize:** int**, nbCores:** int**)***: constructor.

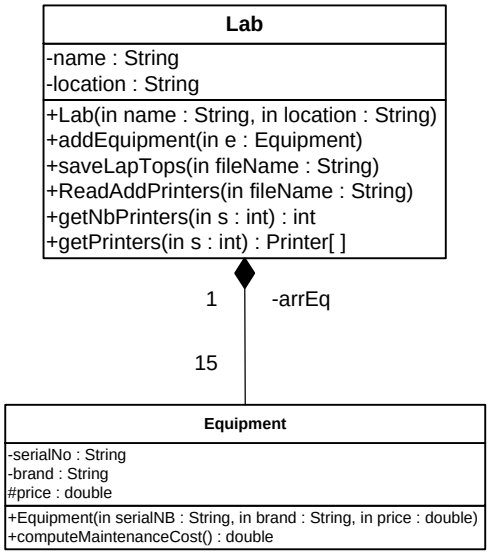
Printer class

- Attributes:
  - ***speed:*** the speed of the printer (number of pages printed per minute).
- Methods:
  - ***Printer (serialNB:** String**, brand:** String**, price:** double**, speed:** int**)***: constructor.
  - ***getSpeed()***: this method returns the speed of the printer.

QUESTION: Translate into Java code the class ***Equipment*** and the class ***Printer***.

Exercise 3:

Let's consider the same class ***Equipment*** described in exercise 1.



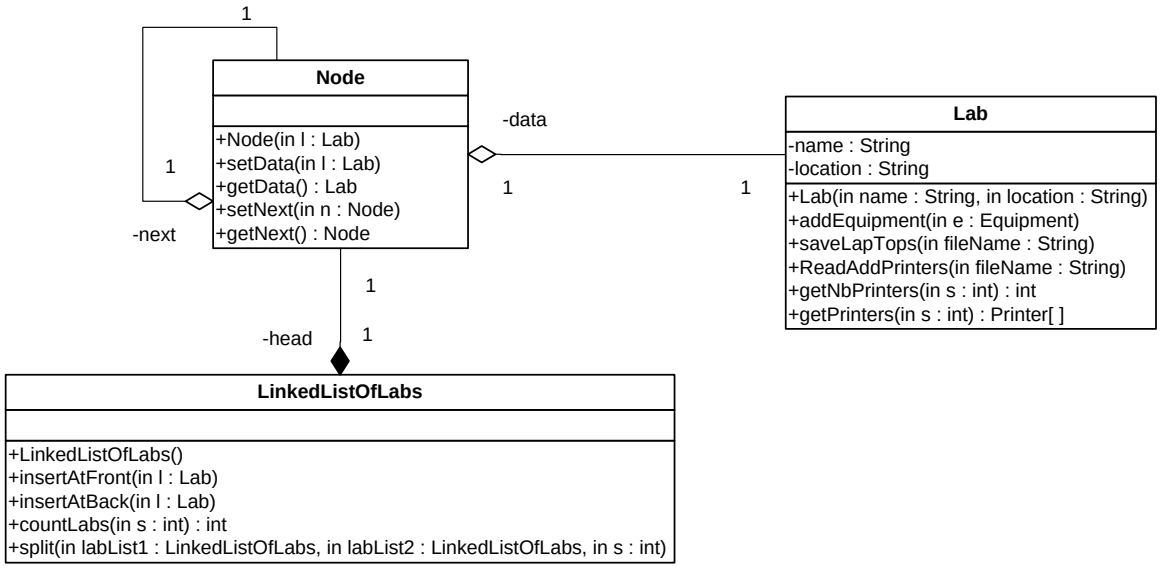
Lab class***:***

- Attributes:
  - ***name***: the name of the Lab.
  - ***location***: the location (room and building numbers) of the Lab.
- Methods:
  - ***Lab(name: String, location: String)***: constructor
  - ***addEquipment (e: Equipment*)*:*** this method adds the equipment ***e*** to the Lab. This method raises an ArrayOutOfBoundException exception if the array ***arrEq*** is full.
  - ***SaveLaptops (fileName: String*)*:*** this method saves all equipment objects of type LapTop in the file which name is passed in the parameter ***fileName***.
  - ***readAddPrinters(fileName: String):*** this method reads equipment objects from the file fileName and adds the read printers to the array ***arrEq*** of the lab. **The different exceptions should be handled separately**.
  - ***getNbPrinters(s: int):*** this method returns the number of printers of the Lab having a speed greater than ***s***. If the parameter ***s*** is less than 15, this method raises an Exception.
  - ***getPrinters(s: int):*** this method returns an array containing all printers of the Lab having a speed greater than ***s***. Make sure that the array should not contain empty cells (the length of the returned array should be exactly equal to the number of the selected printers).

QUESTION: Translate into Java code the class ***Lab.***

Exercise 4:

Let's consider the same class ***Lab*** described in exercise 2.



LinkedListOfLabs class***:***

- Methods:
  - ***LinkedListOfLabs()***: constructor
  - ***insertAtFront (l: Lab*)**: this method adds the Lab ***l*** at the beginning of the list.
  - ***insertAtBack (l: Lab*)**: this method adds the Lab ***l*** at the end of the list.
  - ***countLabs (s: int*)**: this method counts and returns the number of labs having printers with a speed greater than ***s***.
  - ***split(labList1:*** LinkedListOfLabs, ***labList2:*** LinkedListOfLabs, ***s:*** int): This method inserts into ***labList1*** all labs having printers with a speed greater than ***s***. It inserts all other labs into ***labList2.*** The labs of ***labList1*** should be ordered in the same order than the original list. The labs of ***labList2*** should be ordered in the reverse order than the original list.

Exercise 1: -----/9

PersonalComputer object created -----0.5
Desktop 2nd constructor -----0.5
PersonalComputer defalut constructor -----0.5
Desktop 1st constructor -----0.5
PersonalComputer object created -----0.5
Desktop 2nd constructor -----0.5
Iteration 1 -----0.5
Brand: Dell Capacity: 200 -----0.5
Desktop with 3 peripherals -----0.5
With cost = 800 -----0.5
Iteration 2 -----0.5
Exception in Iteration 2 brand is Unknown -----0.5
Iteration 3 -----0.5
Brand: Toshiba Capacity: 15-----0.5
Desktop with 5 peripherals -----0.5
Exception in Iteration 3 Capacity Less Than 20 GB -----0.5
Iteration 4 -----0.5
Exception in Iteration 4 Index out of bound -----0.5

Exercice 2: -----/10

```
public abstract class Equipment implements Serializable { -----1 -----/4
	private String serialNo;
	private String brand;
	protected double price;

	public Equipment(String sn, String b, double p) { -----1
		this.serialNo = sn;
		this.brand = b;
		this.price = p;
	}

	public Equipment(Equipment e) { -----1
		this.serialNo = e.serialNo;
		this.brand = e.brand;
		this.price = e.price;
	}

	public abstract double computeMaintenanceCost();-----1

}

public class Printer extends Equipment { -----1  -----/6
	private int speed;

	public Printer(String sn, String b, double p, int sp) {
		super(sn, b, p); -----1
		this.speed = sp;
	}

	public Printer(Printer p) { -----1
		super(p); -----1
		this.speed = p.speed;
	}

	public double computeMaintenanceCost() { -----1
		return (2*price/100) + speed*100;
	}

	public int getSpeed() { -----1
		return speed;
	}
}
```

Exercice 3: -----/49

```
public class Lab {
	private String name;
	private String location;
	private Equipment arrEq[];  -----1
	private int nbEq;  -----1

	public Lab(String n, String l) {
		this.name = n;
		this.location = l;
		arrEq = new Equipment[15]; -----1
		nbEq = 0;  -----1
	}

	public void addEquipment (Equipment e)
			throws ArrayIndexOutOfBoundsException {  -----1
		if (nbEq >= arrEq.length) throw -----1
			new ArrayIndexOutOfBoundsException("The array is full");

		if (e instanceof Laptop) -----1
			arrEq[nbEq] = new Laptop((Laptop)e); -----1+1
		else
			arrEq[nbEq] = new Printer((Printer)e); -----1+1
		nbEq++; -----1
	}

	public void saveLaptops(String fileName) throws IOException {
		File f = new File(fileName); -----1
		FileOutputStream fo = new FileOutputStream(f); -----1
		ObjectOutputStream objF = new ObjectOutputStream(fo); -----1

		for (int i=0; i < nbEq ; i++) { -----1
			if (arrEq[i] instanceof Laptop) -----1
				objF.writeObject(arrEq[i]); -----1
		}
		objF.close();-----1
	}

	public void readAddPrinters(String fileName)
			throws IOException, ClassNotFoundException {
		File f = new File(fileName);
		FileInputStream fo = new FileInputStream(f); -----1
		ObjectInputStream objF = new ObjectInputStream(fo); -----1

		Equipment e; -----1

		try { -----1
			while (true) { -----1
				e = (Equipment) objF.readObject();-----1
				if (e instanceof Printer) -----1
					addEquipment(e); -----1
			}
		}
		catch(EOFException ex) { -----1
			System.out.println(ex.getMessage());
		}
```

```
            catch (ArrayIndexOutOfBoundsException ex) { -----1
                  System.out.println(ex.getMessage());
            }
            objF.close();-----1
      }

      public int getNbPrinters(int s) throws Exception { -----1
            int n = 0; -----1
            if (s < 15) throw new Exception("Speed is less than 15"); -----1
            for (int i=0; i < nbEq; i++) { -----1
                  if (  arrEq[i] instanceof Printer -----1
                        && ((Printer)arrEq[i]).getSpeed() > s) -----1+1
                        n++; -----1
            }
            return n; -----1
      }

      public Printer[] getPrinters(int s) {
            try { -----1
                  int n = getNbPrinters(s); -----1

                  Printer[] arrP = new Printer[n]; -----1
                  int j = 0; -----1

                  for (int i=0; i < nbEq; i++) { -----1
                        if (  arrEq[i] instanceof Printer -----1
                              && ((Printer)arrEq[i]).getSpeed() > s) -----1
                              arrP[j++] = (Printer)arrEq[i]; -----1
                  }

                  return arrP; -----1
            }
            catch(Exception e) { -----1
                  System.out.println(e.getMessage());
                  return null;
            }
      }
}
```

Exercise 4: -----/22

```
public class LinkedListOfLabs {
	private Node head; -----1

	public LinkedListOfLabs() {
		head = null;
	}

	public void insertAtFront ( Lab m) {

	}

	public void insertAtBack ( Lab m) {

	}

	public int countLabs(int s) {
		int n = 0; -----1
		Node current = head; -----1
		try { -----1
			while (current != null) { -----1
				if (current.getData().getNbPrinters(s) > 0) -----1+1
					n++; -----1
				current = current.getNext();-----1
			}
		}
		catch(Exception e) {System.out.println(e.getMessage());}-----1

		return n; -----1
	}

	public void split(LinkedListOfLabs labList1, LinkedListOfLabs labList2,
				int s) {
		Node current = head; -----1
		try { -----1
			while (current != null) { -----1

				if (current.getData().getNbPrinters(s) > 0) -----1+1
					labList1.insertAtBack(current.getData());---1+1
				else
					labList2.insertAtFront(current.getData());--1+1

				current = current.getNext();---1
			}
		}
		catch(Exception e) {System.out.println(e.getMessage());}---1

	}
}
```