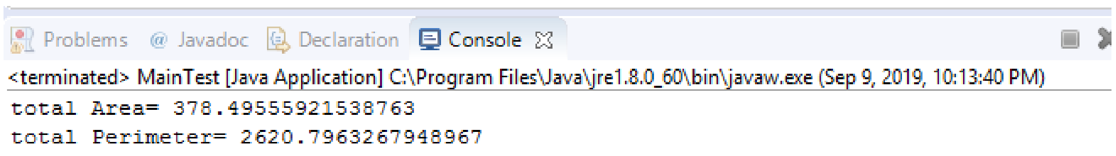
Problem 1

```
package Lesson5_Assignment.problem1;

public class Shape {

	String color;

	public Shape(String color) {
		this.color = color;

	}

	public double calculateArea() {
		return 0.00;

	}

	public double calculatePerimeter() {
		return 0.00;
	}
}

package Lesson5_Assignment.problem1;

public class Circle extends Shape {

	double radius;

	public Circle(double radius, String color) {
		super(color);
		this.radius = radius;
	}

	@Override
	public double calculateArea() {
		return 2 * Math.PI * radius;
	}

	@Override
	public double calculatePerimeter() {
		return Math.PI * radius * radius;
	}
}


package Lesson5_Assignment.problem1;

public class Rectangle extends Shape {
	double width;
	double height;

	public Rectangle(double width, double height, String color) {
		super(color);
		this.width = width;
		this.height = height;
	}

	@Override
	public double calculateArea() {
		return 2 * width + 2 * height;
```

```
      }

      @Override
      public double calculatePerimeter() {
            return width * height;
      }

}

package Lesson5_Assignment.problem1;

public class Square extends Rectangle {

      public Square(double side, String color) {
            super(side, side, color);
      }

}

package Lesson5_Assignment.problem1;

public class MainTest {

      public static void main(String[] args) {

            Shape rec1 = new Rectangle(10, 5, "RED");
            Shape c1 = new Circle(10, "GREEN");
            Shape sq1 = new Square(10, "BLUE");
            Shape c2 = new Circle(20, "ORANGE");
            Shape sq2 = new Square(30, "BROWN");

            Shape[] shapes = { rec1, c1, sq1, c2, sq2 };
            printTotal(shapes);

      }

      public static void printTotal(Shape[] shapes) { // Implement
your code

            double totalArea = 0;
            double totalPerimeter = 0;

            for (Shape shape : shapes) {

                  totalArea += shape.calculateArea();
                  totalPerimeter += shape.calculatePerimeter();
            }
            System.out.println("total Area= " + totalArea);
            System.out.println("total Perimeter= " + totalPerimeter);
      }
}
```

Output



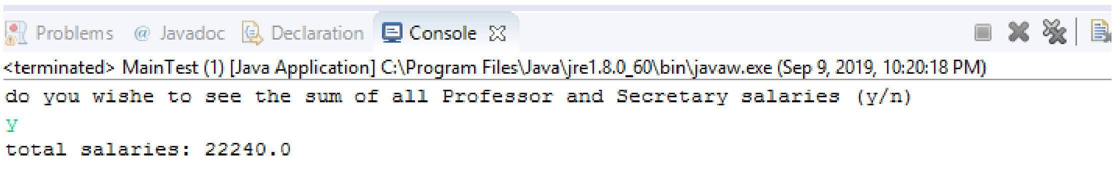
```
<terminated> MainTest [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Sep 9, 2019, 10:13:40 PM)
total Area= 378.49555921538763
total Perimeter= 2620.7963267948967
```

Problem 2

```
package Lesson5_Assignment.problem2;

import java.time.LocalDate;

public class DeptEmployee {

	private String name;
	private LocalDate hireDate;
	protected double salary;

	public DeptEmployee(String name, LocalDate hireDate, double 
salary) {

		this.name = name;
		this.hireDate = hireDate;
		this.salary = salary;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public LocalDate getHireDate() {
		return hireDate;
	}

	public void setHireDate(LocalDate hireDate) {
		this.hireDate = hireDate;
	}

	public double computeSalary() {
		return this.salary;
	}

}

package Lesson5_Assignment.problem2;

import java.time.LocalDate;

public class Professor extends DeptEmployee {

	private int numberOfPublications;

	public Professor(String name, LocalDate hireDate, double 
salary, int numberOfPublications) {
		super(name, hireDate, salary);
		this.numberOfPublications = numberOfPublications;

	}

	public int getNumberOfPublications() {
		return numberOfPublications;
	}
	public void setNumberOfPublications(int numberOfPublications) {
		this.numberOfPublications = numberOfPublications;
	}

}

package Lesson5_Assignment.problem2;

import java.time.LocalDate;

public class Secretary extends DeptEmployee {

	private double overtimeHours;

	public Secretary(String name, LocalDate hireDate, double 
salary, double overtimeHours) {
		super(name, hireDate, salary);
		this.overtimeHours = overtimeHours;
	}

	public double getOvertimeHours() {
		return overtimeHours;
	}

	public void setOvertimeHours(double overtimeHours) {
		this.overtimeHours = overtimeHours;
	}

	@Override
	public double computeSalary() {
		return super.salary + (12 * overtimeHours);
	}

}

package Lesson5_Assignment.problem2;

import java.time.LocalDate;
import java.util.Scanner;

public class MainTest {

	public static void main(String[] args) {
		DeptEmployee professor1 = new Professor("AYA", 
LocalDate.now(), 5000, 5);
		DeptEmployee professor2 = new Professor("AYA", 
LocalDate.now(), 5000, 5);
		DeptEmployee professor3 = new Professor("AYA", 
LocalDate.now(), 5000, 5);

		DeptEmployee secartery1 = new Secretary("YUSSUF", 
LocalDate.now(), 3000, 10);
		DeptEmployee secartery2 = new Secretary("ANS", 
LocalDate.now(), 4000, 10);

		DeptEmployee[] department = new DeptEmployee[5];

		department[0] = professor1;
		department[1] = professor2;
		department[2] = professor3;
		department[3] = secartery1;
```

```
            department[4] = secartery2;

            System.out.println("do you wishe to see the sum of all
Professor and Secretary salaries (y/n) ");

            Scanner scanner = new Scanner(System.in); // Create a
Scanner object
            String answer = scanner.next();
            if (answer.equalsIgnoreCase("y")) {

                  double totalSalary = 0;
                  for (DeptEmployee depEmpolyee : department) {

                        totalSalary += depEmpolyee.computeSalary();
                  }
                  System.out.println("total salaries: "+totalSalary);
            }
            if (answer.equalsIgnoreCase("n")) {
                  System.out.println("you choose no ");
            }

      }
}
```

Output


Problems Javadoc Declaration Console

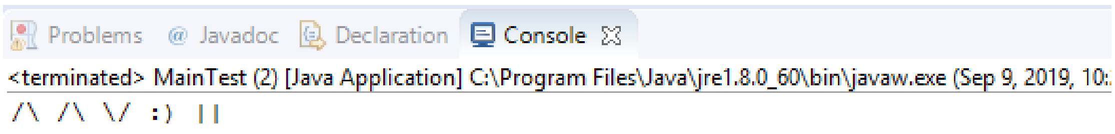
```
<terminated> MainTest (1) [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Sep 9, 2019, 10:20:18 PM)
do you wishe to see the sum of all Professor and Secretary salaries (y/n)
y
total salaries: 22240.0
```

Problem 3

```
package Lesson5_Assignment.problem3;

public interface Figure {

      void getFigure();

}


package Lesson5_Assignment.problem3;

public class DownwardHat implements Figure {

      @Override
      public void getFigure() {
            System.out.print("\\/");
      }

}


package Lesson5_Assignment.problem3;

public class FaceMaker implements Figure {
```

```java
	@Override
	public void getFigure() {
		System.out.print(":)");
	}

}


package Lesson5_Assignment.problem3;

public class UpwardHat implements Figure {

	@Override
	public void getFigure() {
		System.out.print("/\\");
	}

}


package Lesson5_Assignment.problem3;

public class Vertical implements Figure {

	@Override
	public void getFigure() {
		System.out.print("||");
	}

}


package Lesson5_Assignment.problem3;

public class MainTest {

	public static void main(String[] args) {

		Figure upwardHat = new UpwardHat();
		Figure upwardHat2 = new UpwardHat();

		Figure downwardHat = new DownwardHat();
		Figure faceMaker = new FaceMaker();

		Figure vertical = new Vertical();

		Figure[] figures = { upwardHat, upwardHat2, downwardHat,
faceMaker, vertical };

		for (Figure figure : figures) {
			figure.getFigure();
			System.out.print(" ");
		}

	}

}
```

output



```
<terminated> MainTest (2) [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Sep 9, 2019, 10:
/\ /\ \/ :) ||
```

Problem 4

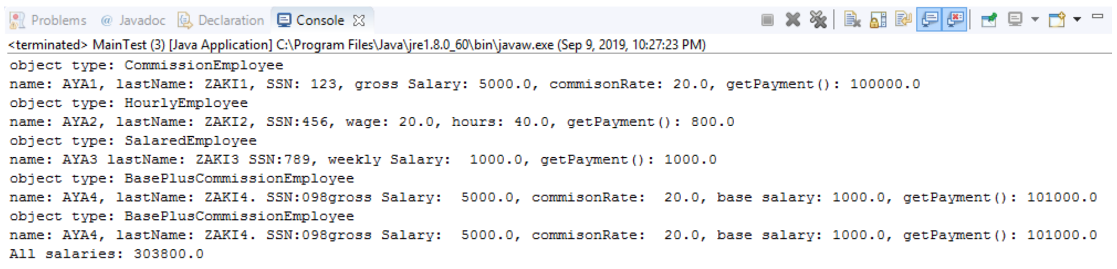
```java
package Lesson5_Assignment.probelm4;

public abstract class Employee {

	private String name;
	private String lastName;
	private String SSN;

	public Employee(String name, String lastName, String SSN) {

		this.name = name;
		this.lastName = lastName;
		this.SSN = SSN;

	}

	public abstract double getPayment();

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getLastName() {
		return lastName;
	}

	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	public String getSSN() {
		return SSN;
	}

	public void setSSN(String sSN) {
		SSN = sSN;
	}

	@Override
	public String toString() {
		return "name: " + name + " " + "lastName: " + lastName +
" " + "SSN: " + " " + SSN;
	}
}
```

```
package Lesson5_Assignment.probelm4;

public class BasePlusCommissionEmployee extends CommissionEmployee {

	private double baseSalary;

	public BasePlusCommissionEmployee(String name, String lastName,
String SSN, double grossSalary,
			double commissionRate, double baseSalary) {
		super(name, lastName, SSN, grossSalary, commissionRate);
		this.baseSalary = baseSalary;
	}

	public double getBaseSalary() {
		return baseSalary;
	}

	public void setBaseSalary(double baseSalary) {
		this.baseSalary = baseSalary;
	}

	@Override
	public double getPayment() {
		return baseSalary + (super.getGrossSalary() *
super.getCommissionRate());
	}

	@Override
	public String toString() {
		return "name: " + super.getName() + ", " + "lastName: " +
super.getLastName() + ". " + "SSN:" + super.getSSN()
				+ "gross Salary: " + " " +
super.getGrossSalary() + ", " + "commisonRate: " + " "
				+ super.getCommissionRate() + ", " + "base
salary: " + baseSalary;
	}

}


package Lesson5_Assignment.probelm4;

public class CommissionEmployee extends Employee {

	private double grossSalary;
	private double commissionRate;

	public CommissionEmployee(String name, String lastName, String
SSN, double grossSalary, double commissionRate) {
		super(name, lastName, SSN);
		this.grossSalary = grossSalary;
		this.commissionRate = commissionRate;
	}

	@Override
	public double getPayment() {
		return grossSalary * commissionRate;
	}

	public double getGrossSalary() {
		return grossSalary;
	}

	public void setGrossSalary(double grossSalary) {
		this.grossSalary = grossSalary;
	}

	public double getCommissionRate() {
		return commissionRate;
	}

	public void setCommissionRate(double commissionRate) {
		this.commissionRate = commissionRate;
	}

	@Override
	public String toString() {
		return "name: " + super.getName() + ", " + "lastName: " +
super.getLastName() + ", " + "SSN:" + " "
				+ super.getSSN() + ", " + "gross Salary: " +
grossSalary + ", " + "commisonRate: " + commissionRate;
	}

}


package Lesson5_Assignment.probelm4;

public class HourlyEmployee extends Employee {

	private double wage;
	private double hours;

	public HourlyEmployee(String name, String lastName, String SSN,
double wage, double hours) {
		super(name, lastName, SSN);
		this.wage = wage;
		this.hours = hours;
	}

	@Override
	public double getPayment() {
		return wage * hours;
	}

	public double getWage() {
		return wage;
	}

	public void setWage(double wage) {
		this.wage = wage;
	}

	public double getHours() {
		return hours;
	}

	public void setHours(double hours) {
		this.hours = hours;
	}
```

```
@Override
	public String toString() {
		return "name: " + super.getName() + ", " + "lastName: " +
super.getLastName() + ", " + "SSN:" + super.getSSN()
				+ ", " + "wage: " + wage + ", " + "hours: " +
hours;
	}

}


package Lesson5_Assignment.probelm4;

public class SalaredEmployee extends Employee {

	private double weeklySalary;

	public SalaredEmployee(String name, String lastName, String
SSN, double weeklySalary) {
		super(name, lastName, SSN);
		this.weeklySalary = weeklySalary;
	}

	@Override
	public double getPayment() {
		return weeklySalary;
	}

	public double getWeeklySalary() {
		return weeklySalary;
	}

	public void setWeeklySalary(double weeklySalary) {
		this.weeklySalary = weeklySalary;
	}

	@Override
	public String toString() {
		return "name: " + super.getName() + " " + "lastName: " +
super.getLastName() + " " + "SSN:"
				+ super.getSSN() + ", " + "weekly Salary: " +
" " + weeklySalary;
	}

}


package Lesson5_Assignment.probelm4;

public class MainTest {

	public static void main(String[] args) {

		Employee commissionEmployee = new
CommissionEmployee("AYA1", "ZAKI1", "123", 5000, 20);
		Employee hourlyEmployee = new HourlyEmployee("AYA2",
"ZAKI2", "456", 20, 40);
		Employee salaredEmployee = new SalaredEmployee("AYA3",
"ZAKI3", "789", 1000);
```

```
		Employee basePlusCommissionEmployee1 = new
BasePlusCommissionEmployee("AYA4", "ZAKI4", "098", 5000, 20, 1000);
		Employee basePlusCommissionEmployee2 = new
BasePlusCommissionEmployee("AYA4", "ZAKI4", "098", 5000, 20, 1000);

		Employee[] employes = { commissionEmployee,
hourlyEmployee, salaredEmployee, basePlusCommissionEmployee1,
				basePlusCommissionEmployee2 };

		double totalSalaries = 0;
		for (Employee employee : employes) {

			System.out.println("object type: " +
employee.getClass().getSimpleName());
			System.out.println(employee.toString() + ", " +
"getPayment(): " + employee.getPayment());
			totalSalaries += employee.getPayment();

		}
		System.out.println("All salaries: " + totalSalaries);
	}

}
```

Output



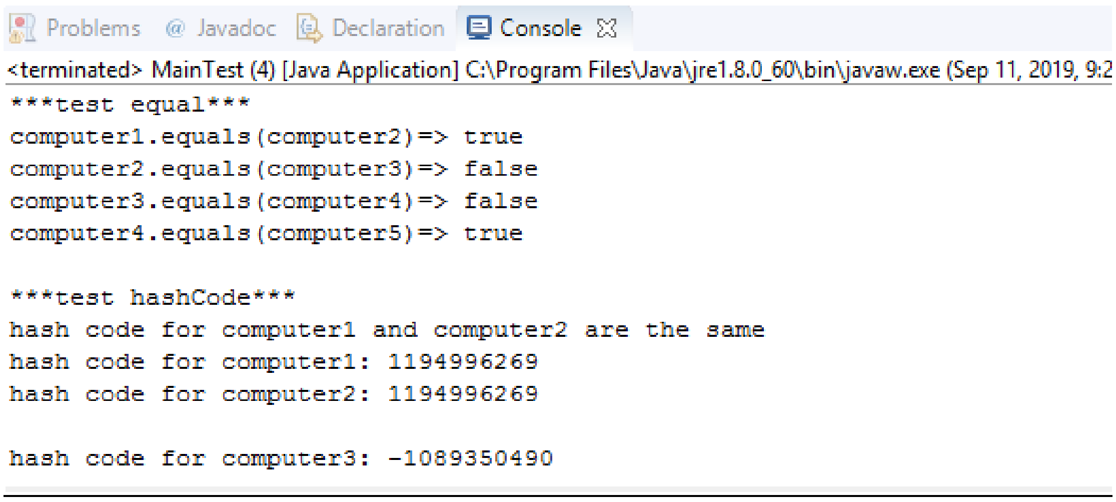
```
<terminated> MainTest (3) [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Sep 9, 2019, 10:27:23 PM)
object type: CommissionEmployee
name: AYA1, lastName: ZAKI1, SSN: 123, gross Salary: 5000.0, commisonRate: 20.0, getPayment(): 100000.0
object type: HourlyEmployee
name: AYA2, lastName: ZAKI2, SSN:456, wage: 20.0, hours: 40.0, getPayment(): 800.0
object type: SalaredEmployee
name: AYA3 lastName: ZAKI3 SSN:789, weekly Salary:  1000.0, getPayment(): 1000.0
object type: BasePlusCommissionEmployee
name: AYA4, lastName: ZAKI4. SSN:098gross Salary:  5000.0, commisonRate:  20.0, base salary: 1000.0, getPayment(): 101000.0
object type: BasePlusCommissionEmployee
name: AYA4, lastName: ZAKI4. SSN:098gross Salary:  5000.0, commisonRate:  20.0, base salary: 1000.0, getPayment(): 101000.0
All salaries: 303800.0
```

Problem 5

```
package Lesson5_Assignment.problem5;

public class Computer {

	private String manufacturer;
	private String processor;
	private int ramSize;
	private double processorSpeed;

	public Computer(String manufacturer, String processor, int
ramSize, double processorSpeed) {
		this.manufacturer = manufacturer;
		this.processor = processor;
		this.ramSize = ramSize;
		this.processorSpeed = processorSpeed;
	}

	public int getRamSize() {
		return ramSize;
	}
	public double getProcessorSpeed() {
		return processorSpeed;
	}

	double computePower() {

		return this.ramSize * this.processorSpeed;

	}

	@Override
	public String toString() {
		return "manufacturer: " + this.manufacturer + " " +
"processor: " + this.processor + " " + "ramSize: " + ramSize
				+ "processorSpeed: " + this.processorSpeed;
	}

	@Override
	public boolean equals(Object obj) {
		if (obj == null)
			return false;

		if (!(obj instanceof Computer))
			return false;

		Computer computer = (Computer) obj;

		return this.manufacturer.equals(computer.manufacturer) &&
this.processor.equals(computer.processor)
				&& this.processorSpeed ==
computer.processorSpeed && this.ramSize == computer.ramSize;
	}

	@Override
	public int hashCode() {
		int result = 17;
		result = 31 * result + processor.hashCode();
		result = 31 * result + manufacturer.hashCode();
		result = 31 * result + (int) (ramSize ^ (ramSize >>>
32));
		long processorSpeedasLong =
Double.doubleToLongBits(processorSpeed);
		result = 31 * result + (int) (processorSpeedasLong ^
(processorSpeedasLong >>> 32));
		return result;
	}

}


package Lesson5_Assignment.problem5;

public class MainTest {

	public static void main(String[] args) {

		Computer computer1 = new Computer("HP", "i5", 500, 50);
		Computer computer2 = new Computer("HP", "i5", 500, 50);
		Computer computer3 = new Computer("DELL", "i5", 500, 50);
		Computer computer4 = new Computer("Mac", "i5", 500, 50);
```

```
            System.out.println("***test equal***");
            System.out.println("computer1.equals(computer2)=> " +
computer1.equals(computer2));// true
            System.out.println("computer2.equals(computer3)=> " +
computer2.equals(computer3));// false
            System.out.println("computer3.equals(computer4)=> " +
computer3.equals(computer4));// false

            Computer computer5 = computer4;
            System.out.println("computer4.equals(computer5)=> " +
computer4.equals(computer5));// true

            System.out.println();
            System.out.println("***test hashCode***");

            System.out.println("hash code for computer1 and computer2
are the same");
            System.out.println("hash code for computer1: " +
computer1.hashCode());
            System.out.println("hash code for computer2: " +
computer2.hashCode());
            System.out.println();

            System.out.println("hash code for computer3: " +
computer3.hashCode());
            System.out.println();

            System.out.println("hashcode for computer4 and computer5
are the same");
            System.out.println("hash code for computer4: " +
computer4.hashCode());
            System.out.println("hash code for computer5: " +
computer5.hashCode());

      }

}
```

Output



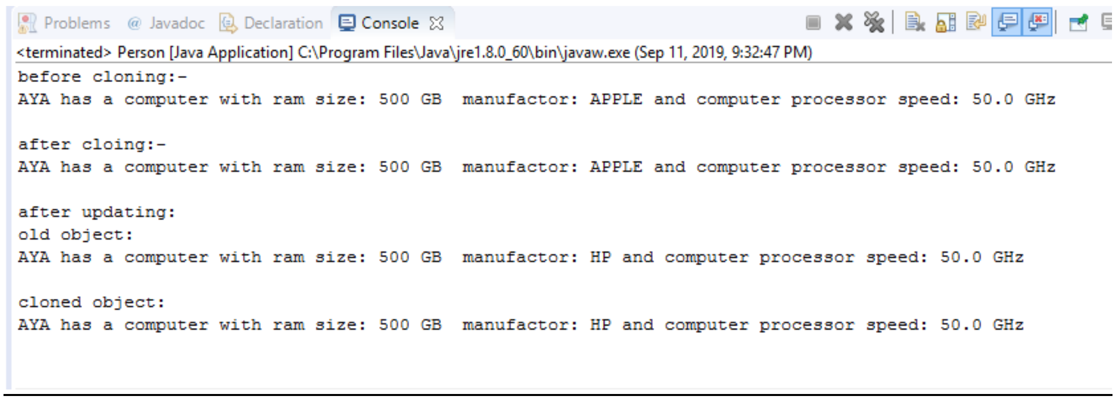
```
<terminated> MainTest (4) [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Sep 11, 2019, 9:2
***test equal***
computer1.equals(computer2)=> true
computer2.equals(computer3)=> false
computer3.equals(computer4)=> false
computer4.equals(computer5)=> true

***test hashCode***
hash code for computer1 and computer2 are the same
hash code for computer1: 1194996269
hash code for computer2: 1194996269

hash code for computer3: -1089350490
```

Problem 6 :- shallow clone

```java
package Lesson5_Assignment.problem6_shallowClone;

public class Computer {

	private String manufacturer;
	private String processor;
	private int ramSize;
	private double processorSpeed;

	public Computer(String manufacturer, String processor, int
ramSize, double processorSpeed) {
		this.manufacturer = manufacturer;
		this.processor = processor;
		this.ramSize = ramSize;
		this.processorSpeed = processorSpeed;
	}

	public String getManufacturer() {
		return manufacturer;
	}


	public void setManufacturer(String manufacturer) {
		this.manufacturer = manufacturer;
	}


	public int getRamSize() {
		return ramSize;
	}

	public double getProcessorSpeed() {
		return processorSpeed;
	}

	double computePower() {

		return this.ramSize * this.processorSpeed;

	}

	@Override
	public String toString() {
		return "manufacturer: " + this.manufacturer + " " +
"processor: " + this.processor + " " + "ramSize: " + ramSize
				+ "processorSpeed: " + this.processorSpeed;
	}

	@Override
	public boolean equals(Object obj) {
		if (obj == null)
			return false;

		if (!(obj instanceof Computer))
			return false;

		Computer computer = (Computer) obj;
		return this.manufacturer.equals(computer.manufacturer) &&
this.processor.equals(computer.processor)
				&& this.processorSpeed ==
computer.processorSpeed && this.ramSize == computer.ramSize;
	}

	@Override
	public int hashCode() {
		int result = 17;
		result = 31 * result + processor.hashCode();
		result = 31 * result + manufacturer.hashCode();
		result = 31 * result + (int) (ramSize ^ (ramSize >>>
32));
		long processorSpeedasLong =
Double.doubleToLongBits(processorSpeed);
		result = 31 * result + (int) (processorSpeedasLong ^
(processorSpeedasLong >>> 32));
		return result;
	}

}


package Lesson5_Assignment.problem6_shallowClone;

public class Person implements Cloneable {

	String name;
	Computer computer;

	public Person() {
	}

	public Person(String name, Computer computer) {
		this.name = name;
		this.computer = computer;

	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public Computer getComputer() {
		return computer;
	}

	public void setComputer(Computer computer) {
		this.computer = computer;
	}

	@Override
	public String toString() {
		return this.name + " has a computer with ram size: " +
this.computer.getRamSize() + " GB " + " manufactor: "
				+ computer.getManufacturer() + " and computer
processor speed: " + this.computer.getProcessorSpeed()
```

```
                        + " GHz";

        }

        @Override
        protected Object clone() throws CloneNotSupportedException {
                Person cloned = (Person) super.clone();
                return cloned;
        }

        public static void main(String[] args) {

                Person originalPerson = new Person("AYA", new
Computer("APPLE", "i7", 500, 50));
                System.out.println("before cloning:-");
                System.out.println(originalPerson);
                System.out.println();

                try {
                        Person clonedPerson = (Person)
originalPerson.clone();
                        System.out.println("after cloing:-");
                        System.out.println(clonedPerson);

                        // update in old object
                        originalPerson.getComputer().setManufacturer("HP");

                        System.out.println();
                        System.out.println("after updating: ");
                        System.out.println("old object:");
                        System.out.println(originalPerson);

                        System.out.println();
                        System.out.println("cloned object:");
                        System.out.println(clonedPerson);

                } catch (CloneNotSupportedException e) {
                        e.printStackTrace();
                }

        }
}
```

Output



```
<terminated> Person [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Sep 11, 2019, 9:32:47 PM)
before cloning:-
AYA has a computer with ram size: 500 GB  manufactor: APPLE and computer processor speed: 50.0 GHz

after cloing:-
AYA has a computer with ram size: 500 GB  manufactor: APPLE and computer processor speed: 50.0 GHz

after updating:
old object:
AYA has a computer with ram size: 500 GB  manufactor: HP and computer processor speed: 50.0 GHz

cloned object:
AYA has a computer with ram size: 500 GB  manufactor: HP and computer processor speed: 50.0 GHz
```

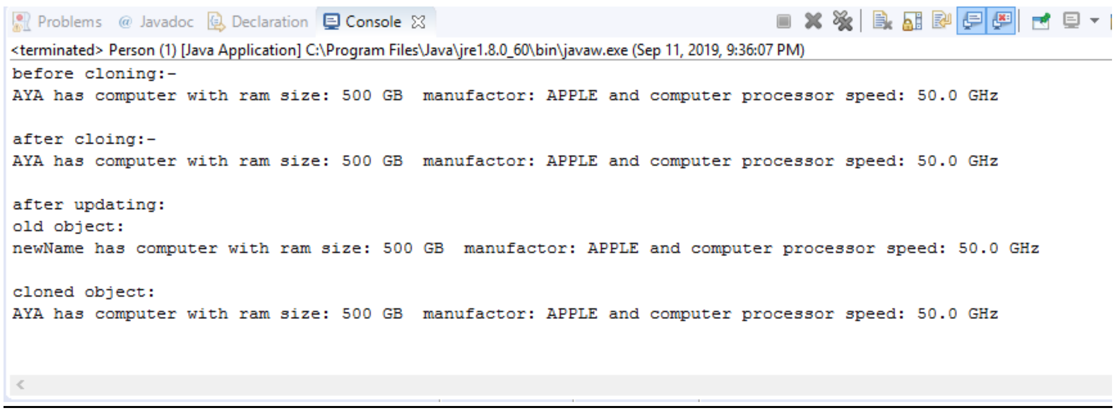
Problem 6:- Deep clone

```
package Lesson5_Assignment.problem6_deepClone;

public class Computer implements Cloneable {

	private String manufacturer;
	private String processor;
	private int ramSize;
	private double processorSpeed;

	public Computer(String manufacturer, String processor, int 
ramSize, double processorSpeed) {
		this.manufacturer = manufacturer;
		this.processor = processor;
		this.ramSize = ramSize;
		this.processorSpeed = processorSpeed;
	}

	public String getManufacturer() {
		return manufacturer;
	}

	public void setManufacturer(String manufacturer) {
		this.manufacturer = manufacturer;
	}

	public int getRamSize() {
		return ramSize;
	}

	public double getProcessorSpeed() {
		return processorSpeed;
	}

	double computePower() {

		return this.ramSize * this.processorSpeed;

	}

	@Override
	public String toString() {
		return "manufacturer: " + this.manufacturer + " " + 
"processor: " + this.processor + " " + "ramSize: " + ramSize
				+ "processorSpeed: " + this.processorSpeed;
	}

	@Override
	public boolean equals(Object obj) {
		if (obj == null)
			return false;

		if (!(obj instanceof Computer))
			return false;

		Computer computer = (Computer) obj;

		return this.manufacturer.equals(computer.manufacturer) && 
this.processor.equals(computer.processor)
```

```
                            && this.processorSpeed ==
computer.processorSpeed && this.ramSize == computer.ramSize;
	}

	@Override
	protected Object clone() throws CloneNotSupportedException {
		Computer clone = (Computer) super.clone();
		return clone;
	}

	@Override
	public int hashCode() {
		int result = 17;
		result = 31 * result + processor.hashCode();
		result = 31 * result + manufacturer.hashCode();
		result = 31 * result + (int) (ramSize ^ (ramSize >>>
32));
		long processorSpeedasLong =
Double.doubleToLongBits(processorSpeed);
		result = 31 * result + (int) (processorSpeedasLong ^
(processorSpeedasLong >>> 32));
		return result;
	}

}


package Lesson5_Assignment.problem6_deepClone;

public class Person implements Cloneable {

	String name;
	Computer computer;

	public Person() {
	}

	public Person(String name, Computer computer) {
		this.name = name;
		this.computer = computer;

	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	@Override
	public String toString() {
		return this.name + " has computer with ram size: " +
this.computer.getRamSize() + " GB " + " manufactor: "
                            + computer.getManufacturer() + " and computer
processor speed: " + this.computer.getProcessorSpeed()
                            + " GHz";

	}
```

```java
	@Override
	protected Object clone() throws CloneNotSupportedException {
		Person cloned = (Person) super.clone();
		cloned.computer = (Computer) this.computer.clone();
		return cloned;
	}

	public static void main(String[] args) {

		Person originalPerson = new Person("AYA", new
Computer("APPLE", "i7", 500, 50));
		System.out.println("before cloning:-");
		System.out.println(originalPerson);
		System.out.println();

		try {
			Person clonedPerson = (Person)
originalPerson.clone();
			System.out.println("after cloing:-");
			System.out.println(clonedPerson);

			// update in old object
			originalPerson.setName("newName");

			System.out.println();
			System.out.println("after updating: ");
			System.out.println("old object:");
			System.out.println(originalPerson);

			System.out.println();
			System.out.println("cloned object:");
			System.out.println(clonedPerson);

		} catch (CloneNotSupportedException e) {
			e.printStackTrace();
		}

	}
}
```

Output



```
<terminated> Person (1) [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Sep 11, 2019, 9:36:07 PM)
before cloning:-
AYA has computer with ram size: 500 GB  manufactor: APPLE and computer processor speed: 50.0 GHz

after cloing:-
AYA has computer with ram size: 500 GB  manufactor: APPLE and computer processor speed: 50.0 GHz

after updating:
old object:
newName has computer with ram size: 500 GB  manufactor: APPLE and computer processor speed: 50.0 GHz

cloned object:
AYA has computer with ram size: 500 GB  manufactor: APPLE and computer processor speed: 50.0 GHz
```