Examen : Ex1 :

```
package examen;

public class Utilisateur {

public String adr;

public Utilisateur [] Amis;

public int nbreAmis;

public Utilisateur(String ad ) {

adr=ad;

nbreAmis=0;

Utilisateur [] Amis= new Utilisateur[100];}

public void ajouterAmi(Utilisateur u) {

if ((nbreAmis<100)&& (u.nbreAmis<100)) {

Amis[nbreAmis]=u;

nbreAmis++;

u.Amis[u.nbreAmis]=this;

u.nbreAmis++;}}

public static boolean sontDesAmis(Utilisateur a, Utilisateur b) {

boolean t=false;

for (int i=0;i<a.nbreAmis;i++) {

if (b.Amis[i].equals(a)) {

t=true;

break;} }

return t; }

public int amis_en_commun(Utilisateur u) {

int commun=0;

for (int i=0;i<nbreAmis;i++) {

for (int j=0;j<u.nbreAmis;j++) {

if (Amis[i].equals(u.Amis[j]))

commun+=1;}}

return commun;}

/*

public int amis_en_commun(Utilisateur u) {

int commun=0;

for (int i=0;i<nbreAmis;i++) {

if(Utilisateur.sontDesAmis(u,Amis[i])

commun++;}

return commun;}

*/

public Utilisateur Ami_le_plus_proche() {

Utilisateur proche = null;

int n=0;

for (int i=0;i<nbreAmis;i++) {

if (n<this.amis_en_commun(Amis[i])) {

n=this.amis_en_commun(Amis[i]);

proche=Amis[i];}}

for (int i=0;i<nbreAmis;i++) {

for (int j=0;j<nbreAmis;j++) {

if(n<this.amis_en_commun(Amis[i].Amis[j])){

n=this.amis_en_commun(Amis[i].Amis[j]);

proche=Amis[i].Amis[j];}}}

return proche;}}
```

Ex2 :

```
package examen;
public abstract class Employe {
private String nom;
private int age;
public Employe(String n, int a) throws AgeException{
nom=n;
if (age<18) throw new AgeException(age);
age=a;}
public abstract double calculerSalaire();//on calcule le salaire selon employé
public String toString() {
return "nom: "+nom+"age: "+age;}}
package examen;
public class Ouvrier extends Employe implements EmployeRisque {
private int nbH;;
private double tarifH;
public Ouvrier(String n, int a, int nb, double t) throws AgeException {
super(n,a);
nbH=nb;
tarifH=t;}
public double calculerPrime() {
return nbH*0.5;}
public double calculerSalaire() {
if(nbH<39)
return nbH*tarifH;
else
return ((nbH-39)*(tarifH*1.3)+(39*tarifH))+calculerPrime();}
public String toString() {
return super.toString()+calculerSalaire()+calculerPrime();}}
package examen;
public class Commercial extends Employe{
private double somFix;
private double chiffreAff;
public Commercial(String n, int a, double sF, double c) throws AgeException {
super(n,a);
somFix=sF;
chiffreAff=c;}
public double calculerSalaire() {
return somFix+(chiffreAff*0.01);}
public String toString() {
return super.toString()+" "+calculerSalaire();}}
package examen;
public class AgeException extends Exception {
private int age;
public AgeException(int age) {
this.age=age;}
public String toString() {
return "AgeException"+this.age;}}
package examen;
public interface EmployeRisque {
public double calculerPrime();}
package examen;
public class Personnel {
private Employe[]tabEmploye;
```

```
private int nb;
public Personnel() {
nb=0;
Employe[]tabEmploye= new Employe[100];}
public void ajouterEmploye(Employe e) {
if(nb<100) {
tabEmploye[nb]=e;
nb++;}}
public void listeEmploye() {
for(int i=0;i<nb;i++) {
System.out.println(tabEmploye[i]);}}
public double salaireMoyen() {
double som=0.0;
for(int i=0;i<=nb;i++) {
som+=tabEmploye[i].calculerSalaire();}
return som/nb;}}
package examen;
public class Test {
public static void main (String[]args) {
try {
Employe e1= new Ouvrier("Ichrak",18,35,160);
Employe e2= new Commercial("dorra",20,36,200);
Personnel p= new Personnel();
p.ajouterEmploye(e1);
p.ajouterEmploye(e2);
p.listeEmploye();
System.out.println(p.salaireMoyen());}
catch (AgeException a) {
System.out.println(a);}}}
```

TP10 : Array List

```
package ex1;
import java.util.ArrayList;
import java.util.Iterator;
public class BoiteEmail {
    private ArrayList<Email>listeEmail=new ArrayList<>();
    public Email rechercherEmail(String ad) {
        Iterator <Email> it= listeEmail.iterator();
        Email m=null;
        while(it.hasNext()) {
            m=it.next();
            if (m.getEnvoyeur().getAdr().equals(ad))
                break;}
        return m;           }
    public void ajouter (Email e) {
        listeEmail.add(e);}
    public void suprrimerEmailLu() {
        for(Email i: listeEmail) {
            if(i.getEtat()==true) {
                listeEmail.remove(i);}}}
    public void lister() {
        for (Email i: listeEmail) {
            if (!i.getEtat())
```

```
                    i.afficher();}
            for (Email i : listeEmail)
                if (i.getEtat())
                    i.afficher();          }
    public void ordreEmail(String o) {
            for (Email i: listeEmail) {
                if(i.getObjet().equals(o)) {
                    System.out.println(listeEmail.indexOf(i));}}  }}
```

```
package ex1;
public class Email {
private String objet;
private Utilisateur envoyeur;
private boolean etat;
public Email(String o, Utilisateur env, boolean i) {
objet=o;
envoyeur=env;
etat=i;}
public boolean getEtat() {
return etat;}
public Utilisateur getEnvoyeur() {
return envoyeur;}
public String getObjet() {
return objet;}
public void modifierEtat() {
if (etat==true)
//if(etat)
etat=false;
else
etat=true;}
public void afficher() {
System.out.println("objet: "+objet+"etat: "+etat+envoyeur.toString());}}
package ex1;
public class Test {
public static void main(String [] args) {
BoiteEmail b= new BoiteEmail();
Utilisateur u1= new Utilisateur ("user1@","ahmed");
Utilisateur u2= new Utilisateur ("user2@","dorra");
Utilisateur u3= new Utilisateur ("user3@","fares");
Email e1= new Email("objet1",u1,true);
Email e2= new Email("objet2",u2,false);
Email e3= new Email("objet3",u3,true);
b.ajouter(e1);
b.ajouter(e3);
b.ajouter(e2);
System.out.println(b.rechercherEmail("user2@"));
b.lister();
b.ordreEmail("objet2");
b.suprrimerEmailLu();}}
package ex1;
public class Utilisateur {
private String adresseEmail;
private String nom;
public Utilisateur(String adr, String n) {
adresseEmail=adr;
```

```
nom=n;}

public String getAdr() {

return adresseEmail;}

public static boolean Comparer(Utilisateur u, Utilisateur u1) {

if ((u.nom.equals(u1.nom) )&& (u.adresseEmail.equals(u1.adresseEmail))){

//u1.equals(u2)

return true;}

else

return false;}

public String toString() {

return "Adresse Email:"+" "+adresseEmail+" "+"Nom:"+" "+nom;}}
```

TP7 : Héritage:

```
package ex1;

public class Fauteuil {

private int prix;

private String tissu;

private String metal;

public Fauteuil(int p,String t,String m) {

prix=p;

tissu=t;

metal=m;}

public void setPrix(int p) {

prix=p;}

public int getPrix() {

return prix;}

public void setTissu(String t) {

tissu=t;}

public String getTissu() {

return tissu;}

public void setMetal(String m) {

metal=m;}

public String getMetal() {

return metal;}

public String toString() {

return "prix:"+" "+prix+" "+"tissu:"+" "+tissu+" "+"metal:"+" "+metal;}}

package ex1;

public class FauteuilElec extends FauteuilRoulant {

private MoteurElecFauteuil M;

public FauteuilElec (int p,String t,String m,RoueAvant r1,RoueAvant r2,RoueArrière
r3,RoueArrière r4,MoteurElecFauteuil M) {

super (p,t,m,r1,r2,r3,r4);

this.M=M;}}

package ex1;

public class FauteuilRoulant extends Fauteuil {

private RoueAvant r1,r2;

private RoueArrière r3,r4;

public FauteuilRoulant(int p,String t,String m,RoueAvant r1,RoueAvant
r2,RoueArrière r3,RoueArrière r4) {

super(p,t,m);

this.r1=r1;

this.r2=r2;

this.r3=r3;

this.r4=r4;}}

package ex1;

public class MoteurElecFauteuil {}
```

```
public class RoueArrière {}
```

```
package ex1;
public class RoueAvant {}
```

```
package ex1;
public class Test {
public static void main (String []args) {
Fauteuil F=new Fauteuil (1500,"cuir","acier");
RoueAvant r1= new RoueAvant();
RoueAvant r2= new RoueAvant();
RoueArrière r3= new RoueArrière();
RoueArrière r4= new RoueArrière();
Fauteuil F1= new FauteuilRoulant(750,"t2","m2",r1,r2,r3,r4);
MoteurElecFauteuil m= new MoteurElecFauteuil();
Fauteuil F2=new FauteuilElec(1000,"T3","M3",r1,r2,r3,r4,m);
Fauteuil [] tab=new Fauteuil[3];
tab[0]=F;
tab[1]=F1;
tab[2]=F2;
float s=0;
for (int i=0;i<3;i++) {
System.out.println(tab[i]);
s+=tab[i].getPrix();}
System.out.println(s); }}
```

Ex2 :

```
package ex3;
public class Barque {
private String nom;
private int num;
public Barque(String n, int N) {
nom=n;
num=N;}
public String getNom() {
return nom;}
public int getNum() {
return num;}
public String toString () {
return "nom:"+" "+nom+"num:"+" "+num;}}
```

```
package ex3;
public class BarqueMoteur extends Barque {
private int puissance;
public BarqueMoteur(String n, int N, int p) {
super(n,N);
puissance=p;}
public int getPuissance() {
return puissance;}
public String toString() {
return super.toString()+" "+"puissance:"+" "+puissance;}}
```

```
package ex3;
public class Port {
private String nomP;
private int capacité;
private float taxe;
Barque []tab;
private int numAct=0;
```

```
public Port(String nP,int c,float t) {
nomP=nP;
capacité=c;
taxe=t;
this.tab=new Barque[capacité];}
public float getTaxe() {
float taxeJ=0;
for (int i=0;i<numAct;i++) {
if(tab[i] instanceof BarqueMoteur) {
if(((BarqueMoteur)tab[i]).getPuissance()>5)
taxeJ+=taxe*1.1f;
else
taxeJ+=taxe*1.2f;}
else
taxeJ+=taxeJ;}
return taxeJ;}
public void enter (Barque b) {
if(numAct<capacité) {
tab[numAct]=b;
numAct++;}}
public void sortir (Barque b) {
for (int i=0;i<numAct;i++) {
if(tab[i].equals(b)) {
int x=i;
for(int j=x;j<numAct;j++) {
tab[j]=tab[j+1];}
tab[numAct]=null;
numAct--;}}}
public String toString() {
String res="";
for (int i=0;i<numAct;i++) {
res+=tab[i].toString()+"\n";}
return res;}}
package ex3;
public class Test {
public static void main (String[]args) {
Barque b=new Barque("b1",502);
Barque bar=new BarqueMoteur("B2",503,100);
Port p1= new Port ("P1",7,132);
p1.enter(b);
p1.enter(bar);
System.out.println(p1);
System.out.println(p1.getTaxe());}}
```

TP8 :

```
package ex1;
public class Cercle implements FormeGéométrique {
private double rayon;
private Point centre;
public Cercle(double r, Point centre) {
rayon=r;
this.centre=centre; }
public double getRayon() {
return rayon;}
public void setRayon(double r) {
```

```
rayon=r;}
public double surface() {
return Math.PI*Math.pow(rayon,2);}
public double perimetre () {
return 2*Math.PI*rayon;} }
package ex1;
public interface FormeGéométrique {
public abstract double surface();
public abstract double perimetre();}
package ex1;
public class Point {
private double ord;
private double abs;
public Point (double abs,double ord) {
this.abs=abs;
this.ord=ord;}}
package ex1;
public class Rectangle implements FormeGéométrique {
private double log;
private double larg;
public Rectangle(double Log,double Larg) {
log=Log;
larg=Larg;}
public void setLong(double L) {
log=L;}
public void setLarg(double l) {
larg=l;}
public double surface() {
return log*larg;}
public double perimetre() {
return (log+larg)*2;}}
package ex1;
public class TestForme {
public static void main (String []args) {
Point P=new Point(8,2);
Cercle c=new Cercle(10,P);
Rectangle R=new Rectangle(3,4);
System.out.println(c.surface()+c.perimetre());
System.out.println(R.surface()+R.perimetre());}}
```

TP9 :

```
package ex1;
public class ConstructionException extends Exception {
private int constVal;
public ConstructionException(int c) {
constVal=c;}
public int getValeur() {
return constVal;}
public String toString() {
return "Construction Exception:"+constVal;}}
package ex1;
public class N {
private int valeurNum;
private final static int Limite=65535;
public N(int v) throws ConstructionException{
```

```
if (v>Limite) throw new ConstructionException(v);//tant qu'il y a throw il y a
throws

else

valeurNum=v;}

public int getValeurNum() {

return valeurNum;}

public static N Produit(N a, N b )throws ResultatException,ConstructionException {

int p=a.valeurNum*b.valeurNum;

if (p>Limite) throw new ResultatException(p,"produit");

return new N(p);}

public static N somme (N a, N b ) throws ConstructionException {

N s =new N(a.valeurNum+b.valeurNum);

return s;}

public static N différence(N a ,N b) throws ConstructionException {

N d=new N(a.valeurNum-b.getValeurNum());

return d;}

public String toString() {

return "valeurNum: "+valeurNum;}}

package ex1;

public class ResultatException extends Exception {

private int x;

private String op;

public int getValeur() {

return x;}

public ResultatException(int c, String o) {

x=c;

op=o;}

public String toString() {

return "Resultat Exception "+" "+op+" "+x;}}

package ex1;

public class Test {

public static void main (String[]args) {

try {

N n1=new N(50);

N n2=new N(60);

System.out.println (N.Produit(n1, n2));

System.out.println(N.somme(n1, n2));

System.out.println(N.différence(n1, n2));}

catch(ConstructionException e) {

System.out.println(e);}

catch(ResultatException e) {

System.out.println(e);}}}
```