Université Libre de Bruxelles Samedi 23 Août 2014

Faculté des Sciences Appliquées

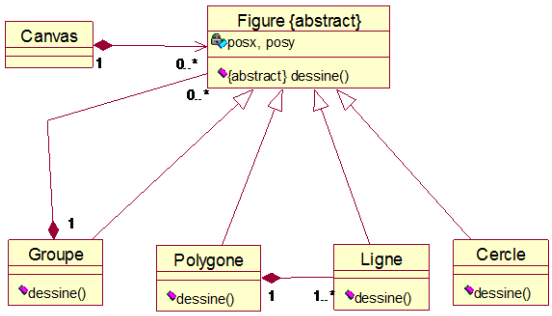
INFO-H-200 : Programmation Orientée Objet

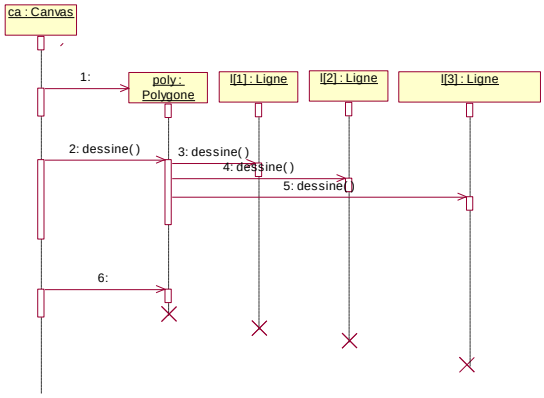
Examen

Remarques préliminaires

- N'oubliez pas d'inscrire nom, prénom et numéro de matricule sur chaque feuille.
- Vous disposez de trois heures et vous ne pouvez pas utiliser de notes.
- Pour les questions où l'on vous demande de justifier ou d'expliquer, la justification ne doit pas faire plus de 4 ou 5 lignes.
- Choisissez bien l'ordre dans lequel vous répondez aux questions : elles n'ont pas toutes le même rapport entre valeur en points et temps nécessaire.
- L'examen est sur 20 points et compte pour 50% de la note finale. Les autres 50% sont donnés par la note du projet.

Question 1 (UML - > Java) : /2





Ecrivez le squelette de code Java qui pourrait être automatiquement généré à partir de ces deux diagrammes UML.

Question 2 (Java) : /2

Expliquez précisément la seule erreur de compilation présente dans les quatre codes qui suivent:

Code 1 :

```
class A {
        private int a;
        public A(int a) {
                this.a = a;
        }
        public void showA() {
                System.out.println(a);
        }
}

public class Test1 {
        public static void main(String[] args) {
                A a = new A();
                a.showA();
        }
}
```

Code 2

```
abstract class A {
        private int a;
        abstract public void showA();
}

class B extends A {
        private int b;
        public B(int b, int a) {
                this.b = b;
        }
}

public class Test3 {
        public static void main(String[] args) {
                B b = new B(3,4);
        }
}
```

Code 3:

```
class A {
        private int a;
        public A(int a) {
                this.a = a;
        }
        public void showA() {
                System.out.println(a);
        }
}

public class Test4 {
        private A a;
        public static void main(String[] args) {
                a = new A(5);
                a.showA();
        }
}
```

Code 4:

```
class A {
        private int a;
        public A(int a) {
                this.a = a;
        }
        public void showA() {
                System.out.println(a);
        }
}

class B extends A {
        private int b;
        public B(int b, int a) {
                this.b = b;
                super(a);
        }
}

public class Test2 {
        public static void main(String[] args) {
                B b = new B(3,4);
        }
}
```

Question 3 (Java) : /4

Retrouvez la sortie console du code sans erreur repris ci-dessous.

```
import java.util.*;

class Arme {
	private String monNom;
	private int maForce;
	public Arme(String nom, int force){
		this.monNom = nom;
		this.maForce = force;
	}
	public String getMonNom(){
		return monNom;
	}
	public int getMaForce(){
		return maForce;
	}
}

abstract class Personnage {
	private String monNom;
	private Arme monArme;
	private int monEnergie, maPosition;
	private Personnage monAmi, monEnnemi;
	private boolean vivant;

	public Personnage(String nom, Arme arme, int energie, int position, Personnage ami,   Personnage ennemi){
		this.monNom = nom;
		this.monArme = arme;
		this.monEnergie = energie;
		this.maPosition = position;
		this.monAmi = ami;
		this.monEnnemi = ennemi;
		this.vivant = true;
	}

	public String getMonNom(){
		return this.monNom;
	}
	public Arme getMonArme() {
		return monArme;
	}
	public int getMonEnergie() {
		return monEnergie;
	}
	public int getMaPosition() {
		return maPosition;
	}
	public void setMaPosition(int maPosition) {
		this.maPosition = maPosition;
	}
	public Personnage getMonAmi() {
		return monAmi;
	}
	public void setMonAmi(Personnage monAmi) {
		this.monAmi = monAmi;
	}
```

```
	public Personnage getMonEnnemi() {
		return monEnnemi;
	}
	public void setMonEnnemi(Personnage monEnnemi) {
		this.monEnnemi = monEnnemi;
	}
	public boolean isVivant() {
		return vivant;
	}
	public int getDistanceAmi(){
		return maPosition-monAmi.getMaPosition();
	}
	public int getDistanceEnnemi(){
		return maPosition-monEnnemi.getMaPosition();
	}
	public abstract void afficheToi();
	public void bougeToi(){
		if(this.getDistanceEnnemi()>0)
			this.maPosition--;
		else
			this.maPosition++;
	}
	public void batsToi(){
		this.afficheCombat();
	}
	public void prendsCa(Arme a){
		if(this.vivant){
			monEnergie-=a.getMaForce();
			if(monEnergie<=0)
				this.jeMeurs(a);
		}
	}
	private void jeMeurs(Arme a){
		this.vivant = false;
		System.out.println("Je suis "+this.monNom+" et je meurs sous les coups d'un(e) "+a.getMonNom());
	}

	private void afficheCombat(){
		System.out.println("Je suis "+this.monNom+" et je frappe "+this.monEnnemi.getMonNom()+" avec mon
		"+this.monArme.getMonNom()+".");
	}
}

class Gentil extends Personnage {
	public Gentil(String nom, Arme arme, int energie, int position, Personnage ami, Personnage ennemi){
		super(nom, arme, energie, position, ami, ennemi);
	}
	@Override
	public void batsToi() {
		if(this.isVivant() && this.getMonEnnemi().isVivant()){
			if(this.getDistanceEnnemi()>1 || this.getDistanceEnnemi()<-1){
				this.bougeToi();
			}
			else{
				this.getMonEnnemi().setMonEnnemi(this);
				this.getMonEnnemi().prendsCa(this.getMonArme());
			}
		} else if(this.isVivant())
			this.bougeToi();
	}
```

```
	@Override
	public void bougeToi() {
		if(Math.abs(this.getDistanceAmi())<Math.abs(this.getDistanceEnnemi()) ||
		!this.getMonEnnemi().isVivant()){
			if(this.getMonAmi().getMonEnnemi().isVivant())
				this.setMonEnnemi(this.getMonAmi().getMonEnnemi());
			else if(!this.getMonEnnemi().isVivant())
			this.setMonEnnemi(this.getMonAmi().getMonAmi().getMonEnnemi());
		}
		super.bougeToi();
	}
	@Override
	public void afficheToi(){
		if(this.isVivant())
			System.out.println("Mon nom est "+this.getMonNom()+" et je suis un Gentil. Mon ennemi est
			"+this.getMonEnnemi().getMonNom());
		else
			System.out.println("Mon nom est "+this.getMonNom()+" et je suis un Gentil mort");
		}
}


class Mechant extends Personnage {
	public Mechant(String nom, Arme arme, int energie, int position, Personnage ami, Personnage ennemi){
		super(nom, arme, energie, position, ami, ennemi);
	}
	@Override
	public void batsToi() {
		if(this.isVivant() && this.getMonEnnemi().isVivant()){
			if(this.getDistanceEnnemi()>1 || this.getDistanceEnnemi()<-1){
				this.bougeToi();
			}
			else{
				this.getMonEnnemi().setMonEnnemi(this);
				this.getMonEnnemi().prendsCa(this.getMonArme());
			}
		}
	}
	@Override
	public void afficheToi(){
		if(this.isVivant())
			System.out.println("Mon nom est "+this.getMonNom()+" et je suis un Méchant. Mon ennemi
			est "+this.getMonEnnemi().getMonNom());
		else
			System.out.println("Mon nom est "+this.getMonNom()+" et je suis un Méchant mort");
	}
}

class Heros extends Gentil {
	private int maRuse;
	public Heros(String nom, Arme arme, int energie, int position, Personnage ami,   Personnage ennemi, int ruse){
		super(nom, arme, energie, position, ami, ennemi);
		if(ruse<1)
			this.maRuse = 1;
		else
			this.maRuse = ruse;
	}
	@Override
	public void batsToi() {
		for(int i=0;i<maRuse;i++){
			super.batsToi();
		}
	}
```

```
	@Override
	public void afficheToi(){
		if(this.isVivant())
			System.out.println("Mon nom est "+this.getMonNom()+" et je suis un Héros. Mon ennemi est
			"+this.getMonEnnemi().getMonNom());
		else
			System.out.println("Mon nom est "+this.getMonNom()+" et je suis un Héros mort");
	}
}

class Volatile extends Mechant {
	public Volatile(String nom, Arme arme, int energie, int position, Personnage ami, Personnage ennemi){
		super(nom, arme, energie, position, ami, ennemi);
	}
	@Override
	public void bougeToi() {
		this.setMaPosition(this.getMonEnnemi().getMaPosition());
	}
	@Override
	public void afficheToi(){
		if(this.isVivant())
			System.out.println("Mon nom est "+this.getMonNom()+" et je suis un Volatile. Mon ennemi est
			"+this.getMonEnnemi().getMonNom());
		else
			System.out.println("Mon nom est "+this.getMonNom()+" et je suis un Volatile mort");
	}
}

public class Epopee {
	public static void main(String[] args){
		ArrayList<Personnage> personnages = new ArrayList<Personnage>();
		personnages.add(new Heros("Thorgal",new Arme("Arc à flèches",3),30,3,null,null,2));
		personnages.add(new Mechant("Orgoff l'Invincible",new Arme("Epée-
		soleil",10),15,9,null,personnages.get(0)));
		personnages.add(new Gentil("Jolan",new Arme("Super Pouvoir",4),15,1,null,null));
		personnages.add(new Volatile("Nixes",new
		Arme("Bec",4),15,30,personnages.get(1),personnages.get(2)));
		personnages.add(new Gentil("Aaricia",new Arme("Charme",2),15,10,personnages.get(0),null));
		personnages.add(new Mechant("Volsung de Nichor",new
		Arme("Epée",3),20,13,personnages.get(1),personnages.get(4)));
		personnages.get(0).setMonAmi(personnages.get(2));
		personnages.get(0).setMonEnnemi(personnages.get(1));
		personnages.get(1).setMonAmi(personnages.get(3));
		personnages.get(2).setMonEnnemi(personnages.get(3));
		personnages.get(2).setMonAmi(personnages.get(4));
		personnages.get(4).setMonEnnemi(personnages.get(5));
		for(int i=0;i<=20;i++){
			for(int j=0;j<personnages.size();j++){
				if(i==0)
					personnages.get(j).afficheToi();
				personnages.get(j).batsToi();
				if(i==20)
					personnages.get(j).afficheToi();
			}
		}
	}
}
```

Question 4 (Java) : /2

Indiquez à même le code les erreurs présentes dans ce code :

```
interface IA {
  public void actionA();
  public void actionC();
}

interface IB {
  public void actionA();
  public void actionB();
}

class A implements IA {
  protected void actionA() {
    System.out.println("je travaille dans A");
  }
  protected void actionA2() {
    actionA();
  }
}

abstract class B extends A implements IA,IB {
  public void actionA() {
    System.out.println("je travaille dans AB");
  }
  public void actionB() {
    System.out.print("hello ");
  }
  public void actionA2() {
    actionB();
  }
  abstract public void actionC();
}

class C extends B {
  public void actionB() {
    System.out.print("Bonjour ");
  }
  public void actionC(){
    actionB();
    actionA();
    actionA2();
  }
}

class D extends A,B {
  public void actionB(){
    System.out.print("Hello ");
  }
  public void actionC(){
    System.out.println("je travaille dans D");
  }
}
```

```
public class Info {
  static IA unA1,unA2, unA3;
  public static void main(String[] args) {
    unA1 = new B();
    unA2 = new C();
    unA3 = new D();
    unA1.actionC();
    unA2.actionC();
    unA3.actionC();
  }
}
```

Question 5 (Conception Java) : /2

Réalisez dans une syntaxe approximative (mais respectant les principes « Orienté Objet ») un code Java contenant deux classes. La première appelée « Complexe », capable de représenter un nombre complexe avec sa partie réelle et imaginaire, elle devra être capable de l'afficher sous sa forme : « a + ib ». Elle devra être capable également de sommer et de soustraire deux nombres complexes. La deuxième classe appelée « Test » sera là simplement pour effectuer quelques tests de bon fonctionnement de la première : la création de deux nombres complexes, leur somme et leur soustraction.

Question 6 (Théorie) : /2

Pourquoi les systèmes de cryptographie sont-ils passés d'une version dite symétrique à une version asymétrique ?

Question 7 (Théorie) : /2

Pourquoi le code d'une instruction élémentaire doit-il reprendre d'une manière ou d'une autre le mode d'adressage utilisé par cette instruction ?

Question 8 (Théorie) : /2

Pour quelle raison la fréquence des « ratés » en mémoire cache doit-elle être idéalement de l'ordre d'un million de fois inférieure à celle des « ratés » en mémoire centrale ?

Question 9 (Théorie) : /2

Expliquez la signification des trois tables des pages représentées ci-dessous. A quoi correspondent les informations données dans les entrées de ces tables ?

0	2	oui
1	4	oui
2	3	oui
3	15	non
4	13	non
5	24	non

0	1	oui
1	5	oui

0	0	oui
1	6	oui
2	37	non
3	28	non