a) A partir du diagramme UML, programmez en Java la classe mère **Compte** et les classes filles **CompteCourant** et **CompteEpargne**

```
public class Compte {

	private final static double SEUIL_SECURITE = -1000 ;

	private int numeroCompte;
	private double solde ;
	private Client proprietaire ;

	public Compte() {
	}

	public Compte(int numeroCompte, double solde, Client proprietaire) {
		this.numeroCompte = numeroCompte;
		this.solde = solde;
		this.proprietaire = proprietaire;
	}

	public void deposer (double montant) {
		if (montant > 0) {
			solde += montant ;
		}
	}

	public boolean retirer (double montant) {
		if (montant < solde) {
			solde = solde - montant ;
			return true ;
		}
		else
		{
			System.out.print("\n Retrait compte numéro: " +
this.numeroCompte + " refusée, le solde de " + proprietaire.getNomClient() + "
n'est pas suffisant");
			return false ;
		}

	}

	/**
	 * Affiche les informations du compte courant
	 */
	public void afficherInformationCompte() {
		System.out.print(" \n Compte numero: " + numeroCompte
				+ ", solde: " + solde + ", Propietaire: " +
proprietaire.getNomClient());
	}
```

```
	/------Accesseurs----------------------------
	public double getSolde() {
		return solde;
	}

	public void setSolde(double solde) {
		if (solde < SEUIL_SECURITE)
			System.out.println ("« ATTENTION tentative d’affectation
suspecte "
					+ "d’un nouveau solde : compte no " +
numeroCompte);
		else
			this.solde = solde;
	}


	public Client getProprietaire() {
		return proprietaire;
	}

	public void setProprietaire(Client proprietaire) {
		this.proprietaire = proprietaire;
	}


	public int getNumero() {
		return numeroCompte;
	}

	public void setNumero(int numeroCompteCourant) {
		this.numeroCompte = numeroCompteCourant;
	}
}
```

```java
 public class CompteCourant extends Compte{

	private double seuilDecouvertAutorise;

	public CompteCourant () {
		super();
	}

	public CompteCourant(int numeroCompte, double solde, Client proprietaire,
double seuilDecouvertAutorise) {
		super(numeroCompte, solde, proprietaire);
		this.SeuilDecouvertAutorise = seuilDecouvertAutorise;
	}

	public boolean retirer (double montant) {
		double nouveauSolde = super.getSolde() - montant ;
		if (nouveauSolde > seuilDecouvertAutorise) {
			super.setSolde(nouveauSolde);
			return true ;
		}
		else
			return false ;
	}

	public void afficherInformationCompte() {
		super.afficherInformationCompte();
		System.out.print(", Seuil Decouverte autorisée: " +
seuilDecouvertAutorise);
	}

	public double getSeuilDecouvertAutorise() {
		return seuilDecouvertAutorise;
	}

	public void setSeuilDecouvertAutorise(double seuilDecouvertAutorise) {
		this.seuilDecouvertAutorise = seuilDecouvertAutorise;
	}
}
```

```java
public class CompteEpargne extends Compte {
	private double tauxInteret;

	public CompteEpargne() {
			super();
		}
	public CompteEpargne(int numeroCompte, double solde, Client proprietaire,
double tauxInteret) {

		super(numeroCompte, solde, proprietaire);
		this.setTauxInteret(tauxInteret);
	}

	public void appliquerInteret()
	{
		 // calcul des intérêts sur le solde
	  double interets = super.getSolde() * tauxInteret;

	  //calcul du nouveau solde
	  double nouveauSolde = super.getSolde() + interets;

	  // modifications du nouveau solde
	  super.setSolde(nouveauSolde);
	}

	public void afficherInformationCompte() {

		super.afficherInformationCompte();
		System.out.print(", Taux d'interet: " + tauxInteret);
	}

	public double getTauxInteret() {
		return tauxInteret;
	}

	public void setTauxInteret(double tauxInteret) {
		this.tauxInteret = tauxInteret;
	}
```

Exercice 2: Polymorphisme

Afin de gagner en généricité, on peut appliquer le même code à des objets de types différents, lorsque les classes de ces objets sont liées par héritage. C'est le principe du **polymorphisme**. Lisez et comprenez le code suivant :

```
public class Banque {

	public static void main(String[] args) {
		//Création des clients
		Client clientJauregui = new Client( "Jauregui" );
		Client clientMasson = new Client( "Masson" );
		Client clientGomez = new Client( "Gomez" );

		//initialisation des comptes
	      Compte c1 = new Compte(1, 100, clientMasson);
	      Compte c2 = new CompteCourant(2, 100, clientJauregui, -1000);
	      Compte c3 = new CompteEpargne(3, 100, clientGomez, 1.5);

	      //affichage des comptes
	      c3.afficherInformationCompte();//CompteEpargne
	      c2.afficherInformationCompte();//CompteCourant
	      c1.afficherInformationCompte();//Compte

	      //appliquer intérêt
	      ((CompteEpargne)c3).appliquerInteret();//CompteEpargne

	      //deposer de l'argent
	      c2.retirer(500); //CompteCourant
	      c3.retirer(500); //CompteEpargne
	      c1.retirer(500); //Compte

	    //affichage des comptes
	      c1.afficherInformationCompte();//Compte
	      c2.afficherInformationCompte();//CompteCourant
	      c3.afficherInformationCompte();//CompteEpargne
	    }
}
```

a) Pour chaque méthode appelée dans le programme précédent, ajoutez des commentaires pour indiquer quel est la classe qui sera utilisée.

```
	//affichage des comptes
		c3.afficherInformationCompte();//CompteEpargne
		c2.afficherInformationCompte();//CompteCourant
		c1.afficherInformationCompte();//Compte

		//appliquer intérêt
		((CompteEpargne)c3).appliquerInteret();//CompteEpargne

		//deposer de l'argent
		c2.retirer(500); //CompteCourant
		c3.retirer(500); //CompteEpargne
		c1.retirer(500); //Compte

	    //affichage des comptes
		c1.afficherInformationCompte();//Compte
		c2.afficherInformationCompte();//CompteCourant
		c3.afficherInformationCompte();//CompteEpargne
```

b) Donnez la sortie du programme précédent.

```
Compte numéro: 3, solde: 100.0, Propriétaire: Gomez, Taux d'intérêt: 1.5
Compte numéro: 2, solde: 100.0, Propriétaire: Jauregui, Seuil Découverte
autorisée: -1000.0
Compte numéro: 1, solde: 100.0, Propriétaire: Masson
Retrait compte numéro: 3 refusée, le solde de Gomez n'est pas suffisant
Retrait compte numéro: 1 refusée, le solde de Masson n'est pas suffisant
Compte numéro: 1, solde: 100.0, Propriétaire: Masson
Compte numéro: 2, solde: -400.0, Propriétaire: Jauregui, Seuil Découverte
autorisée: -1000.0
Compte numéro: 3, solde: 250.0, Propriétaire: Gomez, Taux d'intérêt: 1.5
```

c) Décrivez deux avantages du polymorphisme.

1. Pouvoir référencer des objets sans connaître leur classe véritable au moment de la compilation.
2. Appliquer le même code à des objets de types différents, lorsque les classes de ces objets sont liées par héritage.