Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Jijel
Faculté des Sciences Exactes et Informatique
Département d'informatique

Examen : Environnements et Programmation Dédiés

Nom : ……..... **Master 1 - Intelligence Artificielle**

Prénom : …… **Année universitaire : 2025 - 2026**

Note : ……/**20**. **Durée : 2 h 00**

Situation 1 - QCM - Concepts généraux de programmation : (6 pts)

Chaque question peut avoir **une ou plusieurs** bonnes réponses (**1pt par réponse correcte/2pts réponse n°3**). une réponse **partiellement fausse** vaut 0 point.

1. Soit le programme JavaScript suivant :

```
function f(list, operation) {
   return list
      .map(operation)
      .reduce((acc, valeur) => acc + valeur, 1);
}
const nombres = [1, 2, 3, 4];
const resultat = f(nombres, x => x * 2);
console.log(resultat);
```

☐ Ce programme adopte le **paradigme impératif**, car il définit une séquence d'opérations de filtrage et de transformation sur une collection de données.

☐ Les types des variables de ce programme sont vérifiés au moment de l'exécution, car il est écrit dans un langage **statiquement typé**.

☑ La fonction « **(acc, valeur) => acc + valeur** » passée en argument à **reduce**() est un exemple de fonction de première classe, car elle peut être créée dynamiquement et transmise comme valeur.

☐ f est une **fonction impure**, car elle n'engendre pas un effet de bord et elle respecte le principe de la transparence référentielle.

☐ L'exécution de ce programme affiche la valeur **20** dans la console.

☐ Aucune réponse n'est juste.

2. Parmi les affirmations suivantes concernant les grilles de calcul, cochez celles qui sont correctes :

☑ Les intergiciels (middleware) assurent la coordination et l'interopérabilité dans les grilles.

☐ Toutes les ressources d'une grille appartiennent à une seule entreprise.

☐ Elles utilisent des ressources disponibles à la demande.

☑ Les grilles de calcul permettent l'accès à des ressources distribuées géographiquement.

☐ Aucune réponse n'est juste.

3. Soit le programme java suivant :

```
class ShoppingCart {
   public void addItem(StringBuilder cart, String item)
{
      cart.append(", ").append(item);
   }

   public void addItem(int itemId) {
      System.out.println("Item added with ID: " + itemId);
   }

   public static void main(String[] args) {
      ShoppingCart cartManager = new ShoppingCart();
      StringBuilder myCart = new
StringBuilder("Apple");
      cartManager.addItem(myCart, "Banana");
      cartManager.addItem(101);
      System.out.println("Cart contents: " + myCart);
   }
}
```

☑ La méthode « **addItem** » est une méthode **surchargée**, car elle a le même nom mais des paramètres différents.

☐ La méthode « **addItem** » est une méthode **redéfinie**, car elle a le même nom mais des paramètres différents.

☐ Le type de polymorphisme utilisé dans ce code est le « **polymorphisme paramétrique** ».

☑ Le type de polymorphisme utilisé dans ce code est le « **polymorphisme Adhoc** ».

☐ La valeur de l'objet « **myCart** » après modification reste toujours « **Apple** », car le passage de paramètres par référence n'est pas supporté en java.

☑ La méthode **addItem**(StringBuilder **cart**, String item) montre qu'une modification des propriétés de l'objet « **cart** » affecte l'objet original « **myCart** ».

☐ Aucune réponse n'est juste.

4. Soit la requête HTTP suivante :

```
GET /api/v2/products/15/reviews?limit=10

Host: reviewService.com

Accept: application/json
```

☑ L'URI de cette requête identifie les avis associés au produit dont l'identifiant est 15.

☐ La méthode HTTP GET est utilisée ici pour modifier la liste des avis du produit.

☑ Le paramètre « **limit** » permet de contrôler le nombre de ressources retournées dans la réponse.

☐ L'utilisation de la version « **V2** » dans la requête illustre le principe « **sans état** » (**Stateless**) dans les services REST.

☑ La réponse de cette requête doit être au format **JSON**.

☐ Aucune réponse n'est juste.

5. Lesquels des critères suivants permettent de choisir un langage de programmation de manière **objective**.

☑ « **C** » est un langage compilé offrant un contrôle fin de la mémoire et de bonnes performances, ce qui le rend adapté au développement de systèmes bas niveau.

☐ « **Rust** » est apprécié pour sa syntaxe moderne et élégante, ce qui en fait un langage plus agréable à utiliser que d'autres pour les développeurs.

☑ « **Java** » propose une machine virtuelle garantissant la portabilité du code sur différentes plateformes disposant d'une JVM.

☑ **Python** est un langage interprété disposant d'un vaste écosystème de bibliothèques, facilitant le développement rapide d'applications dans des domaines variés.

☐ Aucune réponse n'est juste.

Situation 2 : 8pts

1 Filtrer les alertes avec severity >= 5 et status != "allowed" 1pts

```
const filteredAlerts = securityLogs.filter( log => log.severity >= 5 && log.status !== "allowed");
```

2 Nombre total de tentatives bloquées 1.5 pts

```
const totalBlockedAttempts = securityLogs
  .filter( log => log.status === "blocked")  0.75 pts
  .reduce( (total, log) => total + log.attempts, 0); 0.75 pts
```

3 Liste des IPs avec attempts > 100 1.5 pts

```
const highRiskIPs = securityLogs
  .filter(log => log.attempts > 100)  0.75 pts
  .map(log => log.origin);  0.75 pts
```

4 Fonction buildIncidentsByService 2 pts

```
function buildIncidentsByService(list) {
  return list.reduce( 0.50 pts
                      (acc, log) =>{
                                        if (!acc.hasOwnProperty(log.service)) { 0.50 pts
                                                acc[log.service] = 0;
                                        }
                                        acc[log.service] += log.attempts; 0.50 pts
                                        return acc;
                      }
                      , {}); 0.50 pts
}
```

5 Fonction generateThreatReport 2 pts

```
function computeRiskScores(log) {
  return log.severity * log.attempts; 0.50 pts
}
function generateThreatReport(list, computeRiskScores) {
  return list.map( 0.50 pts
                    (log) => {
                                        return { id: log.id,  0.50 pts
                                                 riskScore: computeRiskScores(log)  0.50 pts
                                                };
                    });
}
console.log("Threat Report:", generateThreatReport(securityLogs, computeRiskScores));
```

Situation 3 : 3pts

```
public abstract class Espace {    0.50 pts
   protected String spaceId; 0.25 pts
   protected double area; 0.25 pts
  ….
}

public class BureauIndividuel extends Espace { 0.50 pts
   private int nbScreens; 0.25 pts
   private double powerUsage; 0.25 pts
   …..
}

public class SalleReunion extends Espace { 0.25 pts
   private int maxCap; 0.25 pts
   private String videoRes; 0.25 pts
   private boolean hasBoard; 0.25 pts
   ……
}
```

Situation 4 : 3pts

```
public class LivraisonService {

   // 1. Livraison standard (poids seul)
   public double calculerFrais(double poids) { 0.5 pts
      return poids * 2.0;
   }

   // 2. Livraison express (poids + indicateur express)
   public double calculerFrais(double poids, boolean express) { 0.25 pts
      if (express) { 0.25 pts

         return poids * 4.0;
      }
      return calculerFrais(poids);
   }

   // 3. Client premium (poids + type client)
   public double calculerFrais(double poids, String typeClient) { 0.25 pts
      if ("PREMIUM".equalsIgnoreCase(typeClient)) { 0.25 pts
         return poids * 1.5;
      }
      return calculerFrais(poids);
   }

   // 4. Calcul avec poids et distance
   public double calculerFrais(double poids, double distance) { 0.5 pts
      return calculerFrais(poids) + (distance * 0.5); 0.5 pts
   }
}
```

```
public class TestLivraison { 0.5 pts
   public static void main(String[] args) {

      LivraisonService service = new LivraisonService();

      System.out.println("Standard : " + service.calculerFrais(10));

      System.out.println("Express : " +  service.calculerFrais(10, true));

      System.out.println("Premium : " + service.calculerFrais(10, "PREMIUM"));

      System.out.println("Poids + distance : " + service.calculerFrais(10, 50));
   }
}
```