Question 2

1. **La programmation orienté composant** est une approche indépendante de production et de déploiement qui se base d’un fonctionnement modulaire.

Parmi les avantages qui présente la programmation orienté composant, on trouve :

* Réutilisabilité
* Facilité
* Maintenabilité
* Transparence
* Sécurité
* Adaptabilité
* Sous-traitance
* Spécialisation

Un projet informatique collaboré par cette approche changera vers un fonctionnement dynamique qui se compose de plusieurs exécutables partagés par différentes applications géré par un groupe de travail, sa maintenance va devenir fluide et manipulable puisqu’elle va être traité qu’au niveau de couche désiré, ses ressources vont pas créer de soucis puisqu’ils sont pas accessible chez le programmeur, le même projet peut être traité par plusieurs langages de programmation puisqu’il se base sur le concept modulaire, ainsi que les livrables peuvent être découpé, et corriger facilement en cas d’anomalie, du coup on a pas besoin de modifier et livrer l’intégralité du projet mais que le composant mise en œuvre.

1. **La programmation par contrat** est une technique de conception qui se base sur un accord entre un fournisseur et un client. Cette approche permet une amélioration de code à travers une détection de bug qui augmente la robustesse et réutilisation du code.

Parmi les fonctionnalités du paradigme de programmation par contrat :

* Minimiser le temps consacré au débogage du code, en détectant les erreurs lors du développement.
* Faciliter la détection des erreurs.
* Augmenter la réutilisation de code, en s’assurant de sa robustesse.

L’usage de l’approche de programmation par contrat est très important au niveau de développement des applications, dernièrement **GOOGLE** à réussi à introduire ce concept dans un nouveau projet sous le nom de **Cofoja** qui va permettre une sécurité des applications contre les exploits de type dépassement de tampon.

Un autre exemple de **JContractor** qui est une implémentation purement **Java** dont les contrats sont écrits comme des méthodes qui suivent une simple convention d’appellation, **JContractor** fournit une vérification en exécutant les bytecodes à travers des classes qui défissent les contrats, autrement **JContractor** est capable d’ajouter des contrats de vérification en l’introduisant dans les classes et l’exécuter prochainement.

**Références :**

[**https://www.eiffel.com/**](https://www.eiffel.com/)

[**http://jcontractor.sourceforge.net/**](http://jcontractor.sourceforge.net/)

[**https://www.developpez.com**](https://www.developpez.com)