Java & Algorithme – Corrigé du Projet

1 QUESTION 1

Voici le code pour calculer PI selon la méthode proposée à la question 1. Il faut faire varier n pour trouver un résultat plus précis.

```
class Main
{
	public static void main(String args[])
	{
		int n=20;
		int m=0;
		int x, y;
		double pi;

		for(x=0; x<=n; x=x+1)
		{
			for(y=0; y<=n; y=y+1)
			{
				if( (x*x)+(y*y) <= (n*n) )
				{
					m=m+1;
				}
			}
		}

		pi = (4.0*(m-n-1)+1)/(n*n);
		System.out.println(pi);
	}
}
```

2 QUESTION 2

Voici le code pour calculer PI selon la seconde méthode. Il faut faire varier k pour trouver un résultat plus précis :

```
class Main
{
	public static void main(String args[])
	{
		int n=200000;
		int k;
```

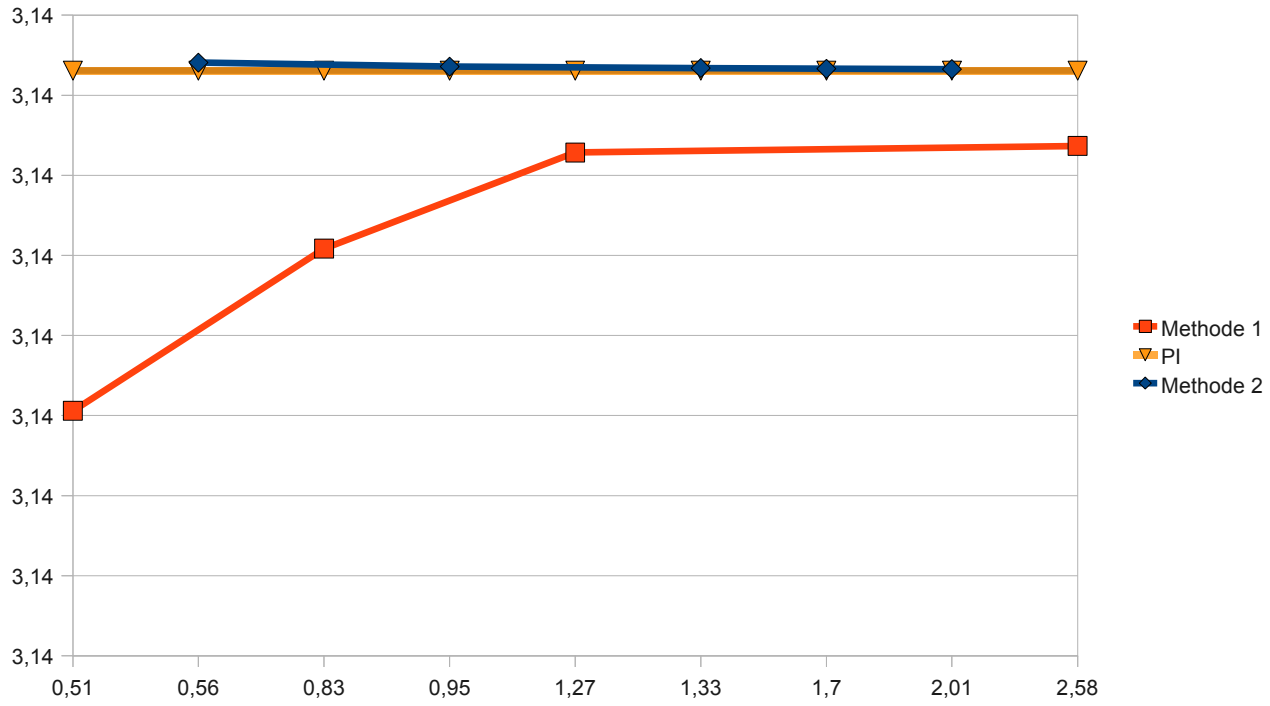
```
            double somme=0;
            int signe=-1;
            double pi;

            for(k=0; k<=n; k=k+1)
            {
                  signe=-signe;
                  somme = somme + signe/(2.0*k+1.0);
            }

            pi = 4.0*somme;
            System.out.println(pi);
      }
}
```

3 QUESTION 3

On procède à une comparaison des deux algorithmes. On regarde quelle valeur les deux algorithmes génèrent en fonction du temps de calcul nécessaire pour trouver la valeur, et on trace une courbe sous Excel (pour la première méthode, on choisira des valeurs de n allant de à , et pour la seconde méthode, on choisira des valeurs de n allant de 10 000 000 à 100 000 000).



En abscisse, le temps de calcul (en seconde), en ordonnée, la valeur générée. En orange, la valeur de Pi.

On voit que la méthode 2 (courbe bleu) donne, pour un même temps de calcul, une valeur plus précise de Pi que la méthode 1 (courbe rouge) (la courbe bleue est presque superposée avec la courbe Pi). La méthode 2 est plus efficace que la méthode 1.