**Ingegneria del Software**

**a.a. 2022-23 Esercizio su STAN4J e Metriche**

**Membri:**

- Gabriele Dellepere (4944557)  
- Kevin Cattaneo (4944382)

**Domande:**

1. Quante sono le dipendenze tra la classe “java.util.BitSet” e la classe “java.util.Arrays”? Di che tipo sono?

Abbiamo identificato quattro dipendenze tra la classe “java.util.BitSet” e la classe “java.util.Arrays”, in particolare:  
- “java.util.BitSet.ensureCapacity(int)”;  
- “java.util.BitSet.toLongArray()”;  
- “java.util.BitSet.trimToSize()”;  
- “java.util.BitSet.valueOf(long[])”.

Queste dipendenze sono afferenti alla classe “java.util.Arrays” in quanto ognuno di questi metodi contiene una chiamata ad “java.util.Arrays.copyOf(long[], int)”.

2. Quante classi con più di 2k ELOC ci sono nel package “java.util”? Riportare le loro dimensioni in ELOC.

Abbiamo identificato tramite una query una sola classe del package “java.util” denominata ““java.util.DualPivotQuicksort” di ELOC superiore a duemila, pari a 2271 righe.

3. Nel package “java.math” è presente un Tangle; da quante componenti è formato?

Il package “java.math” presenta un Tangle formato a cinque componenti, ovvero le classi:

- BigDecimal;  
- SignedMutableBigInteger;  
- BigInteger;  
- BitSieve;  
- MutableBigInteger.

4. Qual è il valore assunto dalla Complessità Ciclomatica (CC) per il metodo “clone()” nella classe “java.util.GregorianCalendar”? Verificare tale valore riportando il calcolo della metrica.

Il valore assunto di CC per il metodo “clone()” è pari a 3.   
Il calcolo della metrica consiste nella somma del numero di punti decisionali (2), rispettivamente alle righe 1946 (if (cdate != null)) e 1947 (if (cdate != gdate)), più uno.

5. Qual è la percentuale di metodi del package “java.util” che hanno Complessità Ciclomatica (CC) maggiore (>) di 4? Qual è la CC più diffusa tra i metodi del package “java.util”?

La percentuale complessiva di metodi del package “java.util” che hanno CC maggiore stretto di quattro è la somma di 2,29% di metodi con CC pari a 5, più 7,6% di metodi con CC maggiore di 5, per un totale di 9,89% dei metodi del package.  
La CC più diffusa tra i metodi del package “java.util” è pari a 1, con una percentuale del 56,8%.

6. Tra i due package “java.util” e “java.math” quale tra i due mostra un miglior bilanciamento tra Instability (I) e Abstractness (A)?

Tra i due package “java.util” e “java.math” si è riscontrato un miglior bilanciamento (nonché la distanza (D)) fra Instability (I) e Abstractness (A) da parte di “java.util”, con una distanza (D) di -0,2 rispetto a -0,38, in quanto più vicina a zero.

7. Qual è il valore della metrica NOC per la classe “java.math.MutableBigInteger”? Verificare tale valore riportando i nomi delle sottoclassi.

Il valore della metrica NOC per la classe “java.math.MutableBigInteger” è pari a uno.   
Per verificare tale valore abbiamo osservato nella sezione “Couplings” di Eclipse (Stan4J) che fra le classi con dipendenza afferente a “java.math.MutableBigInteger”, solo “java.math.SignedMutableBigInteger” ne presenta una di tipo “extends”, determinando la relazione di sottoclasse. Inoltre, abbiamo verificato la presenza dell’estensione visionando (Open Declaration) il codice della classe “java.math.SignedMutableBigInteger” (riga 47).

8. Qual è il valore della metrica RFC per la classe “java.util.PropertyPermission”? Verificare tale valore riportando il numero di metodi della classe + l’elenco dei nomi dei metodi di altre classi chiamati dalla classe stessa.

Il valore della metrica RFC per la classe “java.util.PropertyPermission” è pari a 13.   
Si conta un solo metodo di un’altra classe chiamato dalla classe stessa, ovvero “java.util.PropertyPermission.PermissionCollection()” e 12 metodi propri alla classe, verificati controllando la sezione “Structure Explorer” e il codice sorgente della classe “java.util.PropertyPermission”.

9. Qual è il valore della metrica FAT per la classe “java.util.EventObject”? Verificare tale valore riportando il nome dei nodi, nel grafo delle dipendenze utilizzato per il calcolo di FAT, che hanno almeno un arco entrante.

Il valore della metrica FAT per la classe “java.util.EventObject” è pari a 3.   
Il nome del nodo nel grafo delle dipendenze utilizzato per il calcolo di FAT, che ha almeno un arco entrante, è il nodo di nome “source” (che presenta tre dipendenze afferenti).

10. Qual è il metodo più difficile da testare nel package “java.math”? Quale metrica/metriche ritieni utile/i valutare per rispondere a tale domanda? (argomentare)

Il metodo più difficile da testare nel package “java.math” è “java.math.BigDecimal.(char[],int,int)”, nonché il metodo costruttore, focalizzandosi sull’applicazione della metrica della Complessità Ciclomatica CC, che per tale metodo assume valore pari a 52. Si è scelta questa metrica perché per definizione essa misura il numero di punti decisionali (if, for, while etc.), dai quali dipartono due rami in base al valore della condizione booleana. Questi rami si traducono in almeno due differenti test da effettuare sui due cammini del grafo di esecuzione. Una valida alternativa è anche l’utilizzo della metrica ELOC per stimare il numero di righe di codice all’interno del metodo stesso: in maniera meno precisa si potrebbe affermare che un maggior numero di linee di codice si traduce in un test più complesso. Riteniamo però che quest’ultima affermazione dipenda molto dal contesto, pertanto, la metrica CC si ritiene più realistica: un punto decisionale rappresenta necessariamente una divisione dei flussi di esecuzione da testare.