Per il punto 6 c’è una variabile di classe che non ha il \_ prima del nome, perciò non va bene. Anche se pare che non sia necessario che inizi per forza con il \_, basta che sia minuscola la prima lettera.

private static LocalStringManagerImpl localStrings = new LocalStringManagerImpl(DOLUtils.class);

Riguardo ai punti 13 e 14, dice che dove pratico non dovrebbero superare gli 80 caratteri. Bisognerebbe definire cosa intende per dove pratico, spesso l’intestazione stessa del metodo, comprendendo il nome e i parametri di input, supera gli 80 caratteri, ma lì non c’è niente da fare ed è necessario. Tra l’altro contano anche gli spazi negli 80 caratteri? Io l’unico modo che ho trovato per contarli è selezionare la riga e vedere sotto in brackets quante “colonne” ho selezionato, quindi conta anche gli spazi Per questo io direi di selezionare tutte le righe che superano gli 80, e magari scrivere due nel documento… per fare questo Paolo ci ha consigliato un programmino di ispezione che li segnala… Il rompa dovrebbe avere il file con l’output… L’output l’ho messo su drive, è il file report.txt… fai una ricerca e vedi se tra le righe di codice dei tuoi metodi c’è segnalato qualcosa

È sballato, conta anche se ci sono degli spazi dopo che è finita la riga mi sa. La riga 541 ad esempio mi da 73 caratteri, l’output ne da 81. Conta anche gli spazi che ci sono al posto del tab per l’indentazione, che sicuramente non sono da contare negli 80. Io allora conterei solo gli spazi che ci sono in mezzo alla riga, non quelli prima né quelli dopo, Oooook al max li riconterò dopo xD

29, 30 e 31 mi lasciano un po’ perplesso 29: le variabili devono essere usate. 30: ne abbiamo già discusso, tutti gli oggetti che vengono dichiarati come variabili devono essere istanziate con costruttori. 31: gli oggetti non devono essere null immagino, ma sono cose che te le segnala direttamente eclipse.

Ok allora giusti. Però per il 31 nello Sniffers ho degli oggetti che sono istanziati a null, ma è necessario che sia così, infatti è stato lasciato

Sbaglio o 34 e 35 sono inutili da controllare? Se non fossero corretti il compilatore darebbe errore, quindi sono giusti per forza sisi… per il proper order io e il rompa pensavano tipo a una gerarchia di parametri, tipo prima l’oggetto poi i boolean etc… ma non ha senso perché è a gusto personale (lo abbiamo cercato su internet) quindi si sono corretti

Ok lasciati giusti come avevo messo

Sul 51 sono in dubbio,

ExtendedDeploymentContext context = new DeploymentContextImpl(null, archive, parameters, habitat.<ServerEnvironment>getService(ServerEnvironment.class));

È giusta o sbagliata? :D mi sa che è sbagliata perché la conversione è implicita

Ok segnata come sbagliata

Va che per implicit type conversion intende quella tra interi e double etc non quella che c’è qui, qui è solo una semplice derivazione

Infine, cosa abbiamo intenzione di fare riguardo ai punti di tutta la classe? A parte quelli che sono già stati messi verdi perché vanno bene, gli altri li lasciamo con il punto di domanda e scriviamo qualcosa in una sezione apposita del document? Forse si potrebbe fare così..

Nel 23 ad esempio ho già messo la X però perché è riferito anche ad un issue che c’è nel metodo che dovevamo guardare.

Io comunque lascerei in rosso nella checklist presente nel document quelli riferiti alle classe

Quelli per tutta la classe aveva detto che li avrebbe fatti il Rompa credo… per gli altri due non saprei

Facciamo così, io finisco i metodi che mancano e lollo fa la parte della classe poi finito tutti i punti prendo e faccio una tabella riassuntiva e bom, siamo a cavallo