Qualidade de Software

1. De acordo com a qualidade de produto de software especificada na NBR ISO / IEC 9126, relacione a segunda coluna (sub-característica) com a primeira coluna (característica), e a terceira coluna (definição) com a segunda coluna (sub-característica).

1. Funcionalidade	A. (1) Maturidade	(O) precisão de resultados/efeitos corretos
2. Confiabilidade	B. (6) Instalabilidade	(L) esforço para aprender, aplicar
3. Usabilidade	C. (3) Inteligibilidade	(I) adequação para as tarefas especificadas
4. Eficiência	D. (1) Interoperabilidade	(J) quantidade utilizada de recursos e duração de seu uso
5. Manutenibilidade	E. (5) Analisabilidade	(H) esforço para modificação, adaptação, remoção de defeitos
6. Portabilidade	F. (1) Segurança de acesso	(R) conformidade a normas, convenções, leis, descrições
	G. (6) Capacidade para substituir	(P) risco de efeitos inesperados ocasionados por modificações
	H. (5) Modificabilidade	(K) capacidade em manter desempenho (falhas no software, violação nas interfaces)
	I. (1) Adequação	(T) esforço para validação de modificações
	J. (4) Eficiência em relação aos recursos	(M) desempenho (tempo de resposta, de processamento, e velocidade de execução das funções)
	K. (2) Tolerância a falhas	(D) interagir com sistemas especificados

	L. (3) Facilidade de Aprendizado	(E) esforço para diagnóstico, identificação de falhas
	M. (4) Eficiência em relação ao tempo	(U) recuperação de dados e de desempenho, e de tempo e esforço necessários
	N. (6) Adaptabilidade	(B) esforço para instalação
	O. (4) Precisão	(F) evitar acesso acidental ou deliberado
	P. (5) Estabilidade	(C) esforço para operar, controlar
	Q. (6) Conformidade	(G) esforço e capacidade para substituir outro software
	R. (1) Conformidade	(A) frequência de falhas por defeitos
	S. (3) Operacionalidade	(Q) aderência a convenções e padrões formais de portabilidade
	T. (3) Testabilidade	(N) adaptabilidade a outros ambientes por meios e ações próprias

	U. (2) Recuperabilidade	(S) esforço para entender, identificar

2. Discuta a relação entre os seguintes atributos de qualidade

a. Eficiência e Integridade

A integridade de um software se refere à sua segurança e à garantia de precisão e consistência de suas informações. A eficiência é um atributo que se refere ao nível de desempenho do software. Esses atributos se relacionam pois manter padrões de consistência para as operações de um software é custoso para sua performance em geral. Esses atributos, portanto, não são ortogonais, já que a melhora de um deles acontece em detrimento do outro e vice-versa.

b. Eficiência e Adaptabilidade

Adaptabilidade se refere à possibilidade de um mesmo software ser executado em ambientes diferentes. A eficiência é um atributo que se refere ao nível de desempenho do software. Essas duas características se relacionam porque o uso de um mesmo software em diferentes ambientes pode causar impactos em seu desempenho, já que um alto desempenho pode estar ligado a uma plataforma específica.

3. Como devemos especificar a qualidade de software desejada de tal forma que ela tenha um significado?

Para especificar a qualidade de um software é preciso levar em consideração fatores como expectativas da indústria, necessidade de clientes, requisitos regulatórios etc… Para garantir que esses fatores sejam atendidos é importante desenvolver o software com eles em vista.

4. Defina qualidade interna e externa do software, e discuta a sua relação.

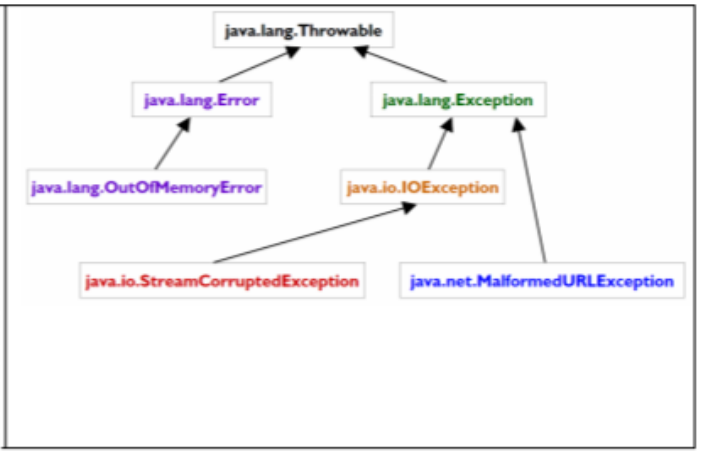
A qualidade interna de um software é a qualidade de seus fatores internos. Os fatores internos de um software são aqueles sob os quais o programador tem controle e pode modificar ou ajustar. Esses fatores incluem características como portabilidade manutenibilidade.

A qualidade externa de um software é aquela que pode ser percebida pelos seus usuários, como sua precisão e sua eficiência. Para o desenvolvimento de um software de qualidade, o mais importante é que ele tenha um boa qualidade externa, porém a qualidade externa é uma consequência da qualidade interna já que os fatores internos são aqueles que estão diretamente sob controle dos desenvolvedores.

Tratamento de Exceções e Assertivas

Para os exercícios 5 a 8, sugere-se primeiramente responder pensando na hierarquia de classes e entendimento sobre tratamento de exceções. Depois, para conferir as respostas, codificar o trecho de código (para isso devem ser criados objetos a priori (como o String s e um ObjectInputStream) e testar o que acontece em tempo de execução.

```
1. try {
2.   URL u = new URL(s); // s is a previously defined String
3.   Object o = in.readObject(); // in is valid ObjectInputStream
4.   System.out.println("Success");
5. }
6. catch (MalformedURLException e) {
7.   System.out.println("Bad URL");
8. }
9. catch (IOException e) {
10.  System.out.println("Bad file contents");
11. }
12. catch (Exception e) {
13.  System.out.println("General exception");
14. }
15. finally {
16.  System.out.println("doing finally part");
17. }
18. System.out.println("Carrying on");
```



5. Que linhas são impressas se os métodos das linhas 2 e 3 completarem com sucesso sem provocar exceções?

A. Success
E. Doing finally part
F. Carrying on

6. Que linhas são impressas se o método da linha 3 provocar um OutOfMemoryError?

E. Doing finally part
F. Carrying on

7. Que linhas são impressas se o método da linha 2 provocar uma MalformedURLException?
B. Bad URL
D. General Exception
E. Doing finally part
F. Carrying on

8. Que linhas são impressas se o método da linha 3 provocar um StreamCorruptedException?

C. Bad File Contents
D. General Exception
E. Doing finally part
F. Carrying on

Projeto de Software Visando o Reuso

9. Desenvolver software visando o reuso sempre traz redução de custos? Justifique sua resposta.

Para desenvolver um software mais rápido e com menos custos é importante adotar um processo baseado no reuso sistemático de componentes de software já

desenvolvidos por outros programadores. Essa prática é exemplificada no uso de bibliotecas, frameworks e outras tecnologias desenvolvidas por terceiros.

Como o software não precisará ser desenvolvido do 0, essa prática irá economizar tempo dos desenvolvedores, e por consequência os recursos que precisarão ser utilizados na implementação do projeto.

10. Faz sentido criar uma classe abstrata que seja final? Justifique sua resposta.

Não faz sentido criar uma classe abstrata como uma classe final porque uma classe abstrata não tem a implementação de seus métodos definida. Uma classe abstrata sempre irá precisar de pelo menos uma classe filha que implemente os métodos que são declarados na classe abstrata.

11. Qual a diferença de sobrecarga (overloading) e sobrescrita (overriding).

Tanto overload quanto override são técnicas utilizadas para o polimorfismo de um código.

A técnica de overload consiste em utilizar o mesmo nome para dois métodos com implementações diferentes. O compilador irá diferenciar os diferentes métodos através de quais as entradas que eles recebem.

A técnica de override acontece quando uma classe herda um método de outra classe, porém o implementa de forma diferente. Logo duas classes filhas de uma mesma classe podem ter um retorno diferente quando chamam o mesmo método presente em sua classe pai.

12. Por que em Java pode-se imprimir diferentes tipos de dados (primitivos e objetos) passando-os como parâmetro do método cujo nome é System.out.println?

Todas as classes Java podem ser imprimidas através do método System.out.println() porque todas elas implementam nativamente o método .toString(). Esse método pode ser modificado pelo desenvolvedor para retornar algum valor específico. Quando ele não é modificado, esse método retorna :

```
getClass (). getName () + '@' + Integer.toHexString (hashCode ())
```

Que é uma string contendo o nome da classe, o caractere ‘@’ , e uma representação do hashCode dessa classe em hexadecimal.

```
public class HelloWorld {
	public String toString() {
		return "HelloWorld!";
	}
}
public class CustomizedHelloWorld extends
HelloWorld {
	private String name;
	public  CustomizedHelloWorld(String
name) {
		this.name = name;
	}
	public String toString() {
		return  "HelloWorld,    "    +
name + "!";
	}
}
```

```
public class  HelloWorldTest {

	public static void main(String[] args) {
		HelloWorld	hw	=	new
CustomizedHelloWorld("TCP");
		new HelloWorldTest().print(hw);
	}

	public void print(HelloWorld helloWorld) {
		System.out.println(helloWorld);
	}

	public   void   print(CustomizedHelloWorld
cHelloWorld) {
		System.out.println("Ola, mundo!");
	}

}
```

13. Executando este código, qual o que é impresso na saída padrão?

a. Hello World!
b. Hello World, TCP!
c. Ola, mundo!

Justifique por que isto ocorre.

Ao executar este código, temos como saída padrão a alternativa b. “Hello World, TCP!”. Isso acontece pois a classe ‘HelloWorldTest’ instancia o objeto ‘hw’ como um ‘HelloWorld’.

```
HelloWorld hw = new CustomizedHelloWorld(“TCP”);
```

Quando a classe ‘HelloWorldTest’ chama o método ‘print’ ela usa como argumento ‘hw’, que é um objeto do tipo ‘HelloWorld’. O compilador irá então chamar o primeiro método descrito, que recebe um objeto do tipo ‘HelloWorld’ como argumento.

```
new HelloWorldTest().print(hw);

public void print(HelloWorld helloWorld) {
System.out.println(helloWorld); }
```

Na classe ‘CostumizedHelloWorld’ o método toString() é sobrescrito o que faz ele retorne ‘ “Hello world” + name + “!” ’. Como a classe ‘hw’ é do tipo ‘CostumizedHelloWorld’, ela irá retornar esta string em seu toString() e isso será exibido pelo método ‘print’ .

14. Usando Java Generics, crie uma classe Tabela parametrizada pelo tipo T, implementada com um array bidimensional de elementos do tipo T. A classe Tabela deve ter um método que dado uma posição x (linha) e y (coluna) deve retorna um objeto do tipo T. Crie uma classe TabelaTest que mostre o uso da classe Tabela.

15. É possível no código da Tabela fazer “new T[lin][col]”, onde lin e col são inteiros? Investigue por que e explique a resposta.

Não é possível criar uma tabela dessa forma pois arrays em java contém em tempo de execução, informações sobre o tipo de seus componentes. Quando criamos um array de tipo genérico “T” , suas informações de tipo são descartadas pelo compilador quando após ele terminar a compilação do código, logo as informações de tipo de “T” estão indisponíveis durante o tempo de execução.