O Design by Contract se trata de uma tecnologia que permite que módulos do sistema estabeleçam uma relação contratual entre si, sendo este caracterizado por especificações e obrigações e benefícios. Ela se base em contratos escritos, ferramentas que verificam se o código satisfaz o que está escrito nos contratos, possibilitando a construção de programas mais confiáveis e reduz a quantidades de bugs.

Para realização dos contratos, usamos recursos da linguagem JML (Java Modeling Language) para especificar o comportamento de tipos (interfaces e classes) e métodos, o que permite por meio de verificações, afirma mais segurança que o sistema esteja correto e ao registro de decisões. Com ela, também podemos usar das especificações como documentação formal do software, evitando qualquer ambiguidade advinda de documentação em linguagem natural. A especificação formal também pode ser usada para gerar para geração de classe de teste. Os Contratos são descritos por uma coleção de assertivas que descrevem precisamente os módulos, são elas a pré-condição (requisito para execução), pós-condição (garantia de saída) e a invariante (afirmação lógica).

Por exemplo podemos citar uma conta bancaria, que para abrir uma conta o valor devera ser maior do que o valor estabelecido (pré-condição) e para depositar o valor deverá ser maior do que zero, para saque ele deve ser menor ou igual ao saldo.

```
package banco;

import  com.google.java.contract. * ;

@Invariants("saldo>0 && saldo==0)
public class Saldo {

	private float saldo;
	private float saque;
	private float depo;

	@Requires("saldo>0||saldo==0")
	@Ensures("x")
	public Saldo(float x) {
		super();
		this.saldo = saldo + x;
	}

	@Requeres("depo>0")
	@Ensures("x")
	public float deposit(float x) {
		this.saldo = saldo + x;
	}

	@Requeres(saque<saldo || saque==saldo)
	@Ensures("x")
	public float saque(float x) {
		this.saldo = saldo - x;
	}

	public float getSaldo() {
		return this.saldo;
	}

}
```