```java
// BadValueException.java
// Implementierung einer eigenen Exception

public class BadValueException extends Exception {
    public int value;
    BadValueException(int value) {
        super("Unzulaessiger Wert:" + value);
        this.value = value;
    }
}

/* Beispiel-Ausgabe:
D:\daten>java BadValueExceptionTest
>> Versuch 1
Bei 10 Muenzen erhaelt jede von 2 Personen ...
... genau 5 Muenzen!

>> Versuch 2
Bei 10 Muenzen erhaelt jede von 4 Personen ...
Ooops: Aufteilung unter 4 klappte nicht: Unzulaessiger Wert:4

>> Versuch 3
Bei 10 Muenzen erhaelt jede von 0 Personen ...
Ooops: Wir haben gar keine Personen zum Verteilen: / by zero

>> Versuch 4
Bei 10 Muenzen erhaelt jede von 2 Personen ...
... genau 5 Muenzen!

>> Versuch 5
Bei 10 Muenzen erhaelt jede von -2 Personen ...
Ooops: Aufteilung unter -2 klappte nicht: Unzulaessiger Wert:-2
*/
```

```java
// BadValueExceptionTest.java
// Java Code Beispiel zur Erzeugung und Behandlung von Exceptions

public class BadValueExceptionTest {
    BadValueExceptionTest() {}

    public static int waehleEineZahl() {
        // liefert eine zufaellige Zahl aus der Menge {-2,0,2,4}
        int a = (int) (Math.random()*4); // a ist aus {0,1,2,3}
        // Zufallszahlen: 0.0 <= Math.random() < 1
        return  (2*a - 2);
    }

    public static int teileMuenzenAuf(int anzMuenzen, int anzPersonen) throws BadValueException {
        // Verteile 'anzMuenzen' Muenzen gerecht unter 'anzPersonen' Personen.

        // bei Division durch 0 tritt eine ArithmeticException (RuntimeException) auf
        int res = anzMuenzen / anzPersonen; // ganzzahlige Division

        // bei negativer Anzahl Personen oder wenn es keine gerechte Verteilung gibt
        // wird eine BadValueException geworfen
        if ((anzPersonen < 0) || (res*anzPersonen != anzMuenzen)) {
            throw(new BadValueException(anzPersonen));
        }

        // wenn alles klappt, wird die Anzahl Muenzen pro Person zurueckgegeben
        return (int) res;
    }

    public static void main(String[] args) {
        // Wähle eine Anzahl von Personen mit waehleEineZahl() und
        // verteile 10 Muenzen gerecht unter ihnen.
        // Wiederhole das ganze 5 Mal.
        int anzMuenzen = 10;
        int anzPersonen;

        for (int i=1; i<=5; i++) {
            anzPersonen = waehleEineZahl();
            System.out.println(">> Versuch " + i);

            try {
                // Im try Block koennen Exceptions auftreten, welche dann
                // in den jeweiligen catch-Anweisungen behandelt werden koennen
                System.out.print("Bei " + anzMuenzen + " Muenzen erhaelt jede von ");
                System.out.println("" + anzPersonen + " Personen ...");
                int res = teileMuenzenAuf(anzMuenzen, anzPersonen);
                System.out.println("... genau " + res + " Muenzen!");

            } catch( ArithmeticException e) {
                // Division durch Null abfangen und melden
                System.out.print("Ooops: Wir haben gar keine Personen ");
                System.out.println("zum Verteilen: " + e.getMessage());

            } catch( BadValueException bve) {
                // BadValueException abfangen und unzulaessige Anzahl Personen melden
                System.out.print("Ooops: Aufteilung unter " + anzPersonen);
                System.out.println(" klappte nicht: " + bve.getMessage());

            } finally {
                // Der finally Block wird immer ausgefuehrt, egal ob im try Block eine
                // Exception auftrat und behandelt wurde durch ein catch oder nicht.
                System.out.println(""); // new line
            }
        }
    }
}
```