```julia
# TensorKit.jl
#
# Main file for module TensorKit, a Julia package for working with
# with tensors, tensor operations and tensor factorizations

module TensorKit

# Exports
#---------
# Types:
export Sector, Irrep, FusionStyle, BraidingStyle
export Abelian, NonAbelian, SimpleNonAbelian, DegenerateNonAbelian,
        SymmetricBraiding, Bosonic, Fermionic, Anyonic # sector properties
export Parity, ZNIrrep, U1Irrep, SU2Irrep, CU1Irrep
        #FermionParity, FermionNumber, FermionSpin # specific sectors
export FibonacciAnyon

export VectorSpace, Field, ElementarySpace, InnerProductSpace, EuclideanSpace #
    abstract vector spaces
export ComplexSpace, CartesianSpace, GeneralSpace, RepresentationSpace, ZNSpace,
    Rep # concrete spaces
export Z2Space, Z3Space, Z4Space, U1Space, CU1Space, SU2Space
export CompositeSpace, ProductSpace # composite spaces
export FusionTree
export IndexSpace, TensorSpace, AbstractTensorMap, AbstractTensor, TensorMap,
    Tensor # tensors and tensor properties
export TruncationScheme
export SpaceMismatch, SectorMismatch, IndexError # error types

# general vector space methods
export space, field, dual, dim, dims, fuse, flip, isdual

# partial order for vector spaces
export infinum, supremum, isisomorphic, ismonomorphic, isepimorphic

# methods for sectors and properties thereof
export sectortype, sectors, hassector, Nsymbol, Fsymbol, Rsymbol, Bsymbol,
        frobeniusschur, twist
export Trivial, ZNSpace, SU2Irrep, U1Irrep, CU1Irrep # Fermion
export fusiontrees, braid, permute#, artin_braid, repartition, insertat, merge

# some unicode
export ⊕, ⊗, ×, ℂ, ℝ, ←, →, ≾, ≿, ≅, ≺, ≻
export ℤ₂, ℤ₃, ℤ₄, U₁, SU₂, CU₁
export ℤ₂Space, ℤ₃Space, ℤ₄Space, U₁Space, CU₁Space, SU₂Space

# tensor maps
export domain, codomain, numind, numout, numin, spacetype, storagetype, eltype
export blocksectors, blockdim, block, blocks

# random methods for constructor
export randuniform, randnormal, randisometry, randhaar

# special purpose constructors
```

```
export zero, one, one!, id, isomorphism, unitary, isometry

# tensor algebra and factorizations
export dot, norm, normalize, normalize!, tr
export mul!, lmul!, rmul!, adjoint!, pinv, axpy!, axpby!
export leftorth, rightorth, leftnull, rightnull,
        leftorth!, rightorth!, leftnull!, rightnull!,
        tsvd!, tsvd, eigen, eigen!, eig, eig!, eigh, eigh!, exp, exp!,
        isposdef, isposdef!, ishermitian, sylvester
export braid!, permute!, transpose, transpose!, twist!
export catdomain, catcodomain

export OrthogonalFactorizationAlgorithm, QR, QRpos, QL, QLpos, LQ, LQpos, RQ,
RQpos,
        SVD, SDD, Polar

# tensor operations
export @tensor, @tensoropt, @ncon, ncon
export scalar, add!, contract!

# truncation schemes
export notrunc, truncerr, truncdim, truncspace, truncbelow

# Imports
#---------
using TupleTools
using TupleTools: StaticLength

using Strided

using TensorOperations: TensorOperations, @tensor, @tensoropt
const TO = TensorOperations

using LRUCache

using HalfIntegers
using WignerSymbols

using Base: @boundscheck, @propagate_inbounds, OneTo, tail, front,
            tuple_type_head, tuple_type_tail, tuple_type_cons,
            SizeUnknown, HasLength, HasShape, IsInfinite, EltypeUnknown, HasEltype
using Base.Iterators: product, filter

import LinearAlgebra
using LinearAlgebra: norm, dot, normalize, normalize!, tr,
                        axpy!, axpby!, lmul!, rmul!, mul!,
                        adjoint, adjoint!, transpose, transpose!,
                        pinv, sylvester,
                        eigen, eigen!, svd, svd!,
                        isposdef, isposdef!, ishermitian,
                        Diagonal, Hermitian
import Base.Meta

const IndexTuple{N} = NTuple{N,Int}
```

```julia
# Auxiliary files
#-----------------
include("auxiliary/auxiliary.jl")
include("auxiliary/dicts.jl")
include("auxiliary/linalg.jl")
include("auxiliary/random.jl")

#--------------------------------------------------------------------
# experiment with different dictionaries
const SectorDict{K,V} = SortedVectorDict{K,V}
const FusionTreeDict{K,V} = Dict{K,V}
#--------------------------------------------------------------------

# Exception types:
#------------------
abstract type TensorException <: Exception end

# Exception type for all errors related to sector mismatch
struct SectorMismatch{S<:Union{Nothing,String}} <: TensorException
    message::S
end
SectorMismatch()=SectorMismatch{Nothing}(nothing)
Base.show(io::IO, ::SectorMismatch{Nothing}) = print(io, "SectorMismatch()")
Base.show(io::IO, e::SectorMismatch) = print(io, "SectorMismatch(", e.message, ")")

# Exception type for all errors related to vector space mismatch
struct SpaceMismatch{S<:Union{Nothing,String}} <: TensorException
    message::S
end
SpaceMismatch()=SpaceMismatch{Nothing}(nothing)
Base.show(io::IO, ::SpaceMismatch{Nothing}) = print(io, "SpaceMismatch()")
Base.show(io::IO, e::SpaceMismatch) = print(io, "SpaceMismatch(", e.message, ")")

# Exception type for all errors related to invalid tensor index specification.
struct IndexError{S<:Union{Nothing,String}} <: TensorException
    message::S
end
IndexError()=IndexError{Nothing}(nothing)
Base.show(io::IO, ::IndexError{Nothing}) = print(io, "IndexError()")
Base.show(io::IO, e::IndexError) = print(io, "IndexError(", e.message, ")")

# Definitions and methods for superselection sectors (quantum numbers)
#----------------------------------------------------------------------
include("sectors/sectors.jl")

# Constructing and manipulating fusion trees and iterators thereof
#------------------------------------------------------------------
include("fusiontrees/fusiontrees.jl")

# Definitions and methods for vector spaces
#-------------------------------------------
include("spaces/vectorspaces.jl")
```

```
# # Definitions and methods for tensors
# #-------------------------------------
# # general definitions
include("tensors/abstracttensor.jl")
include("tensors/tensortreeiterator.jl")
include("tensors/tensor.jl")
include("tensors/adjoint.jl")
include("tensors/linalg.jl")
include("tensors/tensoroperations.jl")
include("tensors/indexmanipulations.jl")
include("tensors/truncation.jl")
include("tensors/factorizations.jl")

end
```