**1: ColonizationSaveGameReader**

1) This class in order to save game information about game data, player data, colony data and colonist.
2) The SaveGame file start with a private static class of GameData. It encodes the data in the game, including mapWidth, mapHeight, numberOfColonies and difficulty.
3) The private static class of PlayerData encodes the data about the player, including newLandName, playerName and humanPlayer.
4) The private static class of ColonyData encodes the data in the colony.
5) The private static class of Colonist encodes the data about the colonist, including occupation, speciality and title.

**2: DesktopEntry**

1) This class in order to add some information about Writer.
2) For different conditions, the results will add different content.
3) Check that the file value from the desktop is empty.
4) If the file starts with FreeColMessage_, it will replace all '-' in the file with '@'.
5) If key.equals("GENERIC_NAME"), the result add the GenericName. If key.equals("COMMENT"), the result add the Comment.
6) If laguageCode is not null, the result adds the '[' + languageCode + ']'.

**3: FlagTest**

1) This class in order to test different flag.
2) The flag file starts with a nine-string flag names.
3) Create JComboBox about FLAG, Decoration, Background, UnionPosition, UnionShape, the number of stars and the number of stripes.
4) Add FLAG, Decoration, Background, UnionPosition, UnionShape, the number of stars and the number of stripes as required.

**4: ForestMaker**

1) This class in order to create forests.
2) The forest file starts with setting the forest data, for example BASE_WIDTH, BASE_HEIGHT and MARGIN etc.
3) It reads the images, and compare the two images.
4) Make space for river by using River_width and River_height.
5) Make space for tree by uging Tree_width and Tree_height.

**5: FSGConverter**

1) This class in order to converts the given input file to an uncompressed and indented XML-file.
2) The FSGConverter file starts with the current FSGConverter monopolize the singletonLock. Then read the file path and written the file path.
3) Convert the file to an XML file based on the path.

**6: GenerateDocumentation**

1) This class in order to generate the document from the given input file.
2) The GenerateDocumentation class starts with Screening the files that meets the criteria.
3) If the source of the reading file is correct, proceed with the try. If the source of the reading file is incorrect, the program will show error reading resources.

**7: InstallerTranslations**

1) First check whether the MAIN_FILE exist. And then check whether the DESTINATION_DIRECTORY is not exists.
2) According to the content of the document to determine the location and information.
3) Convert the file to XML.

**8: ColonizationMapReader**

1) The map file starts with a six-byte header. Byte zero encodes the map width, byte two encodes the map height. The function of the other bytes is unknown, their values, however, are fixed. The header is followed by three "layers", each the size of the map. The first "layer" encodes the terrain type. The function of the other layers is unknown. They are filled with zero bytes.

**9: MapConverter**

1) First check if the file exists. If the file exists, rename it.
2) Try reading the image file. When reading the image file, it will show loaded thumbnail. When the image file cannot be read, it will show no thumbnail present.
3) Save the name and thumbnail of map file.

**10: MergeTranslations**

1) This class in order to merge the target files.
2) Determine if the targetDirectory object is a directory.
3) Tests weather the specified file should be included in a list of dir. When souceFiles is null, it shows that no messages files found in.

**11: RiverMaker**

1) The river file starts with setting the river data, for example BASE_WIDTH, BASE_HEIGHT and MARGIN etc.
2) Create a Point2D object with the given x and y coordinates.
3) Read the river image and generate a 2d rectangle.
4) River branch management

**12: SaveGameValidator**

1) Firstly, look up a factory for the w3 xml schema language, and then compile the schema.
2) Get a validator from the schema.
3) Determine if the file exists, and then determine if the file is directory
4) To validate saveGame File. If success, it will show that successfully validated. If SAXParseException, it will show that which line and which column have problems. If IOException and SAXException, it will show that failed to read.

**13: TranslationReport**

1) This class in order to record file translation errors.
2) First store LanguageStatsRecord by using Arraylist.
3) Determines whether the language file is empty.
4) If missingKeys is not empty, it will show that the number of missingKeys. If copiedFromMaster is not empty, it will show that the number of copiedFromMaster. If missingVariables is not empty, it will show that the number of missingVariables.
5) If superfluouskeys is not empty, it will show that the number of superfluousKeys. If

superfluousVariable is not empty, it will show that the number of superfluousVariable.