```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using CupPlaner.Helpers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CupPlaner.Controllers;

namespace CupPlaner.Helpers.Tests
{
    [TestClass()]
    public class MatchGenerationTests
    {
        CupDBContainer db = new CupDBContainer();

        [TestMethod()]
        public void sletTest()
        {

            Tournament t = db.TournamentSet.Find(1);
            foreach (Division d in t.Divisions.ToList())
            {
                if (d.DivisionTournament != null)
                {
                    foreach (TournamentStage ts in
d.DivisionTournament.TournamentStage.ToList())
                    {
                        foreach (Match m in ts.Matches.ToList())
                        {
                            foreach (Team team in m.Teams.ToList())
                            {
                                team.Matches.Remove(m);
                            }
                            db.MatchSet.Remove(m);
                        }
                        //db.TimeIntervalSet.Remove(ts.TimeInterval);
                        db.TournamentStageSet.Remove(ts);
                    }
                    db.DivisionTournamentSet.Remove(d.DivisionTournament);
                    //dtc.Delete(d.DivisionTournament.Id);
                }

                foreach (Pool pool in d.Pools.ToList())
                {
                    pool.FavoriteFields.Clear();
                    if (pool.IsAuto)
                    {
                        foreach (Team team in pool.Teams)
                        {
                            db.TimeIntervalSet.RemoveRange(team.TimeIntervals);
                        }
                        db.TeamSet.RemoveRange(pool.Teams);

                        db.PoolSet.Remove(pool);
                    }
                }
                db.FinalsLinkSet.RemoveRange(d.FinalsLinks);

                int maxNumOfTeams = 0;

                foreach (Pool p in d.Pools)
                {
                    if (maxNumOfTeams < p.Teams.Count)
                    {
                        maxNumOfTeams = p.Teams.Count;
                    }
                }

                for (int i = 1; i <= maxNumOfTeams; i++)
                {
                    db.FinalsLinkSet.Add(new FinalsLink() { Division = d, PoolPlacement =
i, Finalstage = (i / 2) + 1 });
                }
            }
            db.SaveChanges();
        }

        [TestMethod()]
        public void generateTest()
        {
            MatchGeneration mg = new MatchGeneration();
            mg.Generate(17);
        }

        [TestMethod()]
        public void SameTimeIntervalTest()
        {
            Pool p = new Pool();
            List<DateTime> startDates1 = new List<DateTime>() { DateTime.Parse("16-11-
2015 09:00:00"), DateTime.Parse("17-11-2015 11:00:00"), DateTime.Parse("18-11-2015
08:30:00") };
            List<DateTime> startDates2 = new List<DateTime>() { DateTime.Parse("16-11-
2015 08:00:00"), DateTime.Parse("17-11-2015 12:00:00"), DateTime.Parse("18-11-2015
07:30:00") };
            List<DateTime> startDates3 = new List<DateTime>() { DateTime.Parse("16-11-
2015 07:30:00"), DateTime.Parse("17-11-2015 11:00:00"), DateTime.Parse("18-11-2015
11:30:00") };
            List<DateTime> endDates1 = new List<DateTime>() { DateTime.Parse("16-11-2015
18:00:00"), DateTime.Parse("17-11-2015 22:30:00"), DateTime.Parse("18-11-2015 22:30:00")
};
            List<DateTime> endDates2 = new List<DateTime>() { DateTime.Parse("16-11-2015
20:00:00"), DateTime.Parse("17-11-2015 22:30:00"), DateTime.Parse("18-11-2015 16:00:00")
};
            List<DateTime> endDates3 = new List<DateTime>() { DateTime.Parse("16-11-2015
21:00:00"), DateTime.Parse("17-11-2015 21:30:00"), DateTime.Parse("18-11-2015 22:30:00")
};
            List<TimeInterval> tis1 = new List<TimeInterval>() { new TimeInterval() {
StartTime = startDates1[0], EndTime = endDates1[0] }, new TimeInterval() { StartTime =
startDates1[1], EndTime = endDates1[1] }, new TimeInterval() { StartTime =
startDates1[2], EndTime = endDates1[2] } };
            List<TimeInterval> tis2 = new List<TimeInterval>() { new TimeInterval() {
StartTime = startDates2[0], EndTime = endDates2[0] }, new TimeInterval() { StartTime =
startDates2[1], EndTime = endDates2[1] }, new TimeInterval() { StartTime =
startDates2[2], EndTime = endDates2[2] } };
```

```
            List<TimeInterval> tis3 = new List<TimeInterval>() { new TimeInterval() {
StartTime = startDates3[0], EndTime = endDates3[0] }, new TimeInterval() { StartTime =
startDates3[1], EndTime = endDates3[1] }, new TimeInterval() { StartTime =
startDates3[2], EndTime = endDates3[2] } };


            Team t1 = new Team() { TimeIntervals = tis1 };
            Team t2 = new Team() { TimeIntervals = tis2 };
            Team t3 = new Team() { TimeIntervals = tis3 };

            List<Team> teams = new List<Team>() { t1, t2, t3 };
            MatchGeneration mg = new MatchGeneration();
            List<TimeInterval> ret = mg.SameTimeInterval(new Pool() { Teams = teams,
Division = new Division() { Tournament = new Tournament() { TimeIntervals = tis1 } } });

            Assert.AreEqual(DateTime.Parse("16-11-2015 09:00:00"), ret[0].StartTime);
            Assert.AreEqual(DateTime.Parse("16-11-2015 18:00:00"), ret[0].EndTime);
            Assert.AreEqual(DateTime.Parse("17-11-2015 12:00:00"), ret[1].StartTime);
            Assert.AreEqual(DateTime.Parse("17-11-2015 21:30:00"), ret[1].EndTime);
            Assert.AreEqual(DateTime.Parse("18-11-2015 11:30:00"), ret[2].StartTime);
            Assert.AreEqual(DateTime.Parse("18-11-2015 16:00:00"), ret[2].EndTime);
        }
    }
}
```