GENETSKI ALGORITMI: IZRADA RASPOREDA SATI

ZADATAK: Korištenjem programskog okvira/biblioteke iz prve laboratorijske vježbe, potrebno je napisati program koji pronalazi rješenje sastavljanja rasporeda sati. Raspored sati sastoji se od „blokova" koji u sebi sadrže podatke o nazivu kolegija, nastavniku, dvorani/labosu gdje se nastava održava, trajanju nastave (u satima) i grupi studenata. Raspored sati potrebno je prikazati kao jedan kromosom.
Programsko rješenje najprije mora sadržavati definirane „blokove" koji predstavljaju gene, a također treba definirati i vlastite operatore mutacije/rekombinacije jer postojeći najvjerojatnije ne zadovoljavaju potrebe zadatka.
Nakon toga potrebno je iskoristiti mehanizme iz prve laboratorijske vježbe kako bi se pronašlo konačno rješenje.
Dozvoljeno je korištenje dodatnih klasa za definiranje domenskog modela podataka (podataka o nastavniku, kolegiju, nastavi i sl.)

Zadatak je potrebno riješiti prema sljedećim koracima:

1. Potrebno je uvesti novu klasu koja će predstavljati jedan blok u rasporedu sati, odnosno jedan gen unutar kromosoma koji predstavlja cijeli raspored. U slučaju da ta klasa prestavlja gen, ona mora nasljeđivati klasu „org.jgap.BaseGene" i implementirati sučelja „org.jgap.Gene" i „java.io.Serializable".
   Dozvoljeno je i drugačije definiranje gena unutar kromosoma, tj.gen ne mora biti blok nego nešto drugo, ali moraju biti povezani sa spomenutim blokovima i detaljno opisani u izvještaju. U tom slučaju su i neki od sljedećih koraka drugačiji.

2. Unutar klase iz prvog koraka potrebno je dodati privatne atribute navedene u opisu zadatka. Po potrebi je dozvoljeno dodavati i druge privatne varijable (npr. za potrebe implementacije genetskih operatora, redoslijeda blokova u kromosomu i sl.).

3. Konstruktor klase iz prvog koraka mora primati objekt klase „org.jgap.Configuration" (koji u prvoj naredbi unutar konstruktora predaje konstruktoru nadklase).

4. Kreirati klasu s „main" metodom koja će predstavljati glavni program.

5. Unutar glavnog programa iz četvrtog koraka potrebno je implementirati logiku koja će definirati sve zahtjeve rasporeda sati. Program mora raditi za različite brojeve blokova (počnite s manjim brojevima i „jednostavnijim" zahtjevima rasporeda i postepeno ih povećavajte). Za svaki traženi blok je potrebno

kreirati objekt klase kreirane u prvom koraku.
Klasa koja predstavlja konfiguraciju (prvi parametar konstruktora od klase iz prvog koraka) biti će naknadno kreirana u narednim koracima vježbe.

6. Potrebno je napisati vlastitu implementaciju „Configuration" klase koja unutar sebe ima postavljanje jednog genetskog operatora, za definiranje načina rekombinacije u ovoj implementaciji pomoću operatora „NastavaRecombinationOperator". Primjer implementacije te klase prikazan je u nastavku (**dozvoljeno je i poželjno modificirati detalje te klase**):

```
public class ScheduleConfiguration extends Configuration {
      public ScheduleConfiguration() {
            try {
                  setBreeder(new GABreeder());
                  setRandomGenerator(new StockRandomGenerator());
                  setEventManager(new EventManager());
                  BestChromosomesSelector bestChromsSelector = new
                    BestChromosomesSelector(this, 0.90d);
                  bestChromsSelector.setDoubletteChromosomesAllowed(
                    true);
                  addNaturalSelector(bestChromsSelector, false);
                  setMinimumPopSizePercent(0);
                  setSelectFromPrevGen(1.0d);
                  setKeepPopulationSizeConstant(true);
                  setFitnessEvaluator(new DefaultFitnessEvaluator());
                  setChromosomePool(new ChromosomePool());
                  addGeneticOperator(
                     new MojCustomMadeOperator());
            } catch (InvalidConfigurationException e) {
                  throw new RuntimeException(
                  "Fatal error: DefaultConfiguration class could not
use its " + "own stock configuration values. This should never
happen. " + "Please report this as a bug to the JGAP team.");
            }
      }
}
```

7. Potrebno je kreirati klasu koja će predstavljati „fitness" funkciju za određivanje kvalitete rješenja u traženju ispravnog poretka blokova. Tu klasu je potrebno kreirati na isti način kao i sličnu klasu u prvoj laboratorijskoj vježbi. Preporuka za određivanje iznosa „fitnessa" svakog kromosoma: ne smije biti preklapanja u nastavi unutar iste dvorane, nastavnik ne smije biti u

dvije različite dvorane u isto vrijeme i sl. „Dodatna" svojstva su poželjna, kao npr. da nema velikih rupa u rasporedu između dva bloka i općenito ono što osobno smatrate boljom varijantom rasporeda.

8. Potrebno je kreirati klasu koja će predstavljati vlastiti operator za zamjenu mjesta pojedinih blokova unutar rasporeda. Ta klasa mora implementirati sučelje „GeneticOperator" i implementirati metodu „operate". U toj metodi potrebno je korištenjem „Random" klase i „nextInt" metode odabrati dva nasumična gena unutar kromosoma i zamijeniti njihova mjesta. Kromosom je moguće dohvatiti iz populacije predstavljene ulaznim objektom „a_population" korištenjem metode „getChromosome" koja prima redni broj kromosoma u populaciji. Kromosom kojem je promijenjen raspored gena potrebno je prebaciti u listu koja je predana kao drugi parametar metode, pod nazivom „a_candidateChromosomes".

9. Prema uzoru na definiranje parametara i dodavanje genetskih operatora u objektu konfiguracije iz prve laboratorijske vježbe, slično je potrebno napraviti i s konfiguracijskim objektom iz šestog koraka nakon njegovog kreiranja u glavnom programu.

10. Kako se zbog složenije strukture gena više ne može kreirati populacija automatskim postupkom, to je potrebno napraviti s vlastitom metodom kako bi se ta populacija mogla koristiti u zadanom broju evolucija. Pritom je potrebno pripaziti da se svaki kromosom, odnosno geni unutar njega koji predstavljaju segmente pjesme „izmiješaju" na odgovarajući način. Primjer implementacije metode koja generira „random" populaciju prikazan je u nastavku:

```
private static Population
generateRandomPopulation(ScheduleConfiguration conf,
      int numberOfSegments, Schedule[] scheduleSegments) throws
      InvalidConfigurationException {

      IChromosome[] chromosomes = new Chromosome[POPULATION_SIZE];
      for (int i = 0; i < POPULATION_SIZE; i++) {
         Shedule[] genes = new Shedule[numberOfSegments];
         List<Shedule> scheduleList =
          Arrays.asList(scheduleSegments);
         Collections.shuffle(scheduleList);

          int n = 0;

         for(Schedule segment : scheduleList) {
           genes[n] = segment;
```

```
            n++;
        }

        Chromosome newChromosome = new Chromosome(conf, genes);
        chromosomes[i] = newChromosome;
    }
    Population population = new Population(conf, chromosomes);
    return population;
}
```

11. Slično kao i u prvoj laboratorijskoj vježbi, nakon generiranja „random" populacije pozivom metode iz prošlog koraka potrebno je definirati „sample" kromosom i kreirati objekt koji predstavlja populaciju.

12. Na kraju je još potrebno provesti zadani broj evolucija pomoću kojih će genetski algoritam definiranim operatorom (ili operatorima) tražiti konačno rješenje rasporeda. Ispis koraka pretraživanja je također potrebno ispisivati u konzolu kao i kod prve laboratorijske vježbe.

13. Napisati izvještaj o rezultatima izvođenja **prema uzoru na prvu laboratorijsku vježbu**. Potrebno je napisati zasebnu tablicu s rezultatima za svaku korištenu konfiguraciju, grafički prikazati mjerenja te na kraju napisati zaključak zbog čega određena konfiguracija genetskog algoritma daje najbolje, odnosno najlošije rezultate.

UPUTE:

1. Koristiti prvu laboratorijsku vježbu za zadržavanje slične strukture implementacije rješenja.
2. Za bolje razumijevanje korištenja klasa iz programskog okvira „jGAP" koristiti tutorial i link objavljen u sklopu druge laboratorijske vježbe na lms.tvz.hr. U tutorialu potražiti poglavlje o Creating Custom Genes i Creating Custom Genetic Operators. http://www.atetric.com/atetric/javadoc/cn.apiclub.third/jgap/3.6.2/org/jgap/impl/
3. Za dobivanje boljih rezultata potrebno je proširivati ili mijenjati konfiguraciju genetskog algoritama (kao u prvoj laboratorijskoj vježbi).
4. Zadatak mora pronalaziti rješenja za jednostavnije rasporede (npr. 2 nastavnika, svako predavanje traje 1h, za samo jednu grupu studenata,…) Za dodatne bodove, potrebno je zakomplicirati raspored dodatnim zahtjevima, većim brojem nastavnika/dvorana/grupa itd.