#### Лабораторная работа 1

**Темы**: ***Основы программирования на Java.***

***Обработка событий. Механизм делегирования событий.***

#### Практические задания

1. Познакомиться с особенностями технологии Java и изучить синтаксис языка Java.
2. Изучить основные понятия и термины обработки событий в модели делегирования событий на Java.
3. Разобрать приведенные примеры **FontsList**, **LinesDraw, KeyCodes, Timer**, **DrawFig**.
4. Разработать программу. Основная задача – разработка упрощенной имитации поведения объектов (все последующие лабораторные работы будут расширять это задание). Объекты реализуются через наследование: абстрактный класс + интерфейс → наследники.

Рабочий цикл программы:

* запускается процесс симуляции по клавише, генерируются объекты классов согласно заданию;
* симуляция завершается по другой клавише, выводится статистическая информация.

1. Для решения задачи:

* Разработать абстрактный класс объекта, согласно варианту индивидуального задания.
* Создать интерфейс IBehaviour, задающий поведение объекта (далее будут реализоваться алгоритмы движения объектов в окне программы).
* Реализовать иерархию классов, определяющих объекты по варианту и реализующие интерфейс IBehaviour.
* Создать интерфейс AbstractFactory для создания объектов классов.
* Реализовать от него класс ConcreteFactory, возвращающий нужный объект, согласно варианту индивидуального задания.
* По желанию можно использовать паттерн Factory Method.
* Создать класс Habitat (среда), определяющий размер рабочей области и хранящий массив объектов, с параметрами заданными вариантом. Предусмотреть в классе метод Update, вызывающийся по таймеру и получающий на вход время, прошедшее от начала симуляции. В данном методе должны генерироваться новые объекты и помещаться в поле визуализации в случайном месте. Визуализация объекта – схематично, плюсом будет, если объект будет похож на оригинал (можно использовать готовые небольшие картинки);

Рабочее окно программы – область визуализации среды обитания объектов;

1. Симуляция должна запускаться по клавише **B** и останавливаться по клавише **E**. При остановке симуляции список уничтожается. Время симуляции должно отображаться текстом в области визуализации и скрываться/показываться по клавише **T**;
2. По завершению симуляции в поле визуализации должна выводиться информация о количестве и типе сгенерированных объектов, а также время симуляции. Текст должен быть форматирован, т.е. выводиться с использованием разных шрифтов и цветов.
3. Параметры симуляции задаются в классе Habitat.
4. При необходимости или желанию можно использовать другие паттерны.

***Вариант 1***

Объект – муравей. Бывают 2 видов: рабочий и воин. Рабочие рождаются каждые N1 секунд с вероятностью P1. Воины рождаются каждые N2 секунд с вероятностью P2.

***Вариант 2***

Объект – пчела. Бывают 2 видов: трутень и рабочий. Трутни рождаются каждые N1 секунд, если их количество менее K% от общего числа пчел, в противном случае – не рождаются вовсе. Рабочие рождаются каждые N2 секунд с вероятностью P.

***Вариант 3***

Объект – аквариумная рыбка. Бывают 2 видов: золотая и гуппи. Золотые рыбки рождаются каждые N1 секунд с вероятностью P1. Гуппи рождаются каждые N2 секунд с вероятностью P2.

***Вариант 4***

Объект – кролик. Бывают 2 видов: обыкновенный и альбинос. Обыкновенные кролики рождаются каждые N1 секунд с вероятностью P1. Альбиносы рождаются каждые N2 секунд, при условии, что их количество менее K% от общего числа кроликов, в противном случае – не рождаются вовсе.

***Вариант 5***

Список объектов продажи на автомобильном рынке состоит из 2-х видов машин: грузовые и легковые. Грузовые машины генерируются каждые N1 секунд с вероятностью P1. Легковые генерируются каждые N2 секунд с вероятностью P2.

***Вариант 6***

Рабочий коллектив компании состоит из разработчиков и менеджеров. Разработчики генерируются каждые N1 секунд с вероятностью P1. Менеджеры генерируются каждые N2 секунд при условии, что их количество менее K% от общего числа разработчиков, в противном случае – не генерируются.

***Вариант 7***

Список жилых домов города состоит из двух типов: капитальный, деревянный. Капитальные дома генерируются каждые N1 секунд с вероятностью P1. Деревянные дома генерируются каждые N2 секунд с вероятностью P2.

***Вариант 8***

Список транспортных средств на дороге состоит из двух категорий: автомобили и мотоциклы. Автомобили генерируются каждые N1 секунд с вероятностью P1. Мотоциклы генерируются каждые N2 секунд с вероятностью P2.

***Вариант 9***

Объект обучающийся. Бывает 2 видов: студент (муж. пола) и студентка (жен. пола). Студенты генерируются каждые N1 секунд с вероятностью P1. Студентки генерируются каждые N2 секунд с вероятностью P2.

***Вариант 10***

Картотека налоговой инспекции состоит из записей двух типов: физические и юридические лица. Физические лица генерируются каждые N1 секунд с вероятностью P1. Юридические лица генерируются каждые N2 секунд с вероятностью P2.

***Вариант 11***

Объекты – домашние животные. Бывают 2 видов: кошки и собаки. Кошки генерируются каждые N1 секунд с вероятностью P1. Собаки генерируются каждые N2 секунд с вероятностью P2.

***Вариант 12***

Объекты – птицы. Бывают 2 видов: птенцы и взрослые птицы. Взрослые птицы генерируются каждые N1 секунд с вероятностью P1. Птенцы генерируются каждые N2 секунд при условии, что их количество менее K% от общего числа взрослых птиц, в противном случае – не генерируются.

***Вопросы для подготовки к защите***

1. Что такое первичный класс приложения? Какой обязательный метод он должен содержать?
2. Какие существуют виды переменных Java, чем они отличаются друг от друга?
3. Какие примитивные типы определены в Java?
4. Что делает конструктор класса? Должен ли он обязательно явно присутствовать в объявлении класса?
5. Какие существуют виды ссылочных типов? Как реализуются ссылочные переменные?
6. Что такое типы, определенные пользователем?
7. В чем особенности строковых переменных?
8. Чем массивы Java отличаются от массивов других языков, их преимущества?
9. Как переменные различных видов передаются в качестве параметров методам?
10. Что такое элементы класса и элементы экземпляра класса, чем они отличаются друг от друга? Как нужно указывать, что переменная или метод является элементом класса, а не экземпляра?
11. Для чего используются модификаторы доступа? Какие существуют модификаторы доступа, как они ограничивают доступ к элементам?
12. Что позволяет делать процесс наследования? Что такое суперкласс и подкласс?
13. Что такое повторное использование кода?
14. Какие заранее определенные переменные содержит каждый класс Java?
15. Что можно сделать при помощи переменной this?
16. Что можно сделать при помощи переменной super?
17. Что такое скрытие переменной, затемнение переменной и замещение метода?
18. Как импортировать классы из пакетов?
19. Как добавить класс в пакет?
20. Как ускорить вывод графических изображений, загружаемых из файла и устранить мерцание при выводе изображений?
21. Объект какого типа выполняет все графические операции?
22. Почему рекомендуется использовать ограниченный набор цветов?
23. Почему рекомендуется использовать ограниченный набор шрифтов?
24. Как получить список всех доступных шрифтов?
25. Что такое событие? Когда возникают события?
26. Что такое объект слушателя?
27. Какие существуют способы создания слушателя события?
28. Какие методы отвечают за обработку простых событий от мыши?
29. Какие методы отвечают за обработку простых событий от клавиатуры?
30. Что такое модель делегирования событий?
31. Как подключиться к прослушиванию события?
32. Что такое вложенные классы?
33. Что такое анонимные классы?

#### Лабораторная работа 2

**Темы**: ***Разработка графического интерфейса приложения.***

#### Практические задания

1. Познакомиться с основными графическими библиотеками Java - AWT и Swing и их основными компонентами. Изучить классы менеджеров компоновки, классы создания меню приложения, основных и диалоговых окон и т.д.
2. Для хранения генерируемых объектов использовать динамический массив объектов. Для обеспечения доступа всем элементам приложения и создания только одного экземпляра массива объектов используйте структурный паттерн Singleton.
3. При реализации шаблона Singleton используется класс Singleton. В этом классе определяется закрытый конструктор, имеется закрытая статическая переменная, в которой хранится ссылка на единственный экземпляр данного класса, а также определен статический метод доступа, возвращающий ссылку на этот экземпляр. Так же класс содержит объект коллекции и открытые методы доступа к ней. Таким образом можно в любом месте приложения получить ссылку на объект класса Singleton и доступ к коллекции объектов.
4. Доработать программу, созданную в лабораторной работе № 1:
   1. поделить рабочую область окна приложения на 2 части. Визуализация переносится в одну часть окна, панель управления в другую;
   2. добавить кнопки «Старт» и «Стоп» в панель управления. Они должны запускать и останавливать симуляцию соответственно. Если симуляция остановлена, то кнопка «Стоп» должна блокироваться. Если симуляция идет, то блокируется кнопка «Старт». Клавиши **B** и **E** должны функционировать по-прежнему;
   3. добавить переключатель «Показывать информацию», который разрешает отображение модального диалога из 7 пункта задания;
   4. добавить группу из 2 исключающих переключателей: «Показывать время симуляции» и «Скрывать время симуляции». Клавиша **T** должна функционировать по-прежнему;
   5. используя различные менеджеры компоновки, реализовать интерфейс пользователя согласно индивидуальному заданию;
   6. добавить в программу главное в меню и панель инструментов, в которых продублировать основные команды вашего интерфейса пользователя;
   7. при остановке симуляции должно появляться модальное диалоговое окно (при условии, что оно разрешено) с информацией о количестве и типе сгенерированных объектов, а также времени симуляции. Вся информация выводится в элементе TextArea, недоступном для редактирования. В диалоговом окне должно быть 2 кнопки: «ОК» и «Отмена». При нажатии на «ОК» симуляции останавливается, а при нажатии на «Отмена», соответственно продолжается;
   8. предусмотреть проверку данных вводимых пользователем. При вводе неверного значения обрабатывать исключительную ситуацию: выставлять значение по умолчанию и выводить диалоговое окно с сообщением об ошибке;
   9. Реализовать следующие элементы управления:
   * Периоды рождения объектов – текстовые поля;
   * Для задания вероятностей рождения объектов комбобокс и список (шаг значений 10%);
   * Дополнить интерфейс поясняющими метками.

***Вопросы для подготовки к защите***

1. Что такое GUI?
2. Какие графические библиотеки есть в Java?
3. Какой принцип работы компонентов AWT? Недостатки библиотеки.
4. Какой принцип работы компонентов Swing?
5. Что такое элементы управления и что такое контейнеры?
6. Какие классы элементов управления существуют?
7. Что необходимо сделать, чтобы добавить компонент в контейнер?
8. Как можно перехватить и обработать события, пришедшие от компонентов?
9. Какие типы переключателей существуют?
10. Как несколько переключателей объединить в группу?
11. Чем отличаются выпадающие и невыпадающие списки? Как осуществляется в них выбор элементов?
12. Что такое текстовые поля и текстовые области? Чем они отличаются?
13. Что такое контейнеры? Какие основные виды контейнеров существует?
14. Для чего чаще всего используются панели?
15. В чем основное отличие окон и панелей?
16. Что является обязательным параметром конструктора при создании экземпляра класса окон?
17. Каковы отличительные особенности имеют фреймы?
18. Как добавить меню в контейнер? Как реализуется вложенное меню?
19. Как создать новое меню и добавить в него элементы?
20. Какими методами обрабатываются события меню?
21. Для чего в основном используются окна диалогов?
22. Каковы важные отличия окон диалогов от фреймов?
23. Объект какого класса должен обязательно быть родителем диалогового окна?
24. Что такое модальное окно?
25. Как создать диалог своего класса?
26. Для чего предназначены менеджеры компоновки? Какие существуют режимы размещения?

#### Лабораторная работа 3

**Темы**: ***Классы-коллекции.***

#### Практические задания

1. Изучить особенности реализации классов-коллекций в Java.
2. Доработать программу, созданную в лабораторной работе № 2:
3. добавить генерируемым объектам понятия «время рождения» и «время жизни». Время рождения устанавливается в момент генерации объекта, и по значению соответствует времени, прошедшему от начала симуляции. Время жизни – время, через которое объект должен исчезнуть, считая от времени рождения;
4. вынести установку параметров времени жизни объектов в пользовательский интерфейс. Для каждого типа объекта должно задаваться собственное время. Рекомендуется использовать текстовые поля, но следуют помнить о проверке на ввод некорректных данных;
5. добавить генерируемым объектам уникальные целочисленные идентификаторы (случайные числа), которые назначаются при генерации объекта. Для хранения сгенерированных идентификаторов используйте коллекцию удобную для поиска по варианту;
6. использовать коллекции по варианту. При генерации объекта происходит добавление его в коллекцию (в класс добавить поле идентификатора), а во вторую коллекцию: идентификаторы существующих объектов, в третью идентификатор + время рождения. При возникновении события по таймеру обойдите коллекцию и удалите все объекты, время жизни которых истекло, а также все данные во вспомогательных коллекциях; Для обеспечения доступа к коллекциям используйте структурный паттерн Singleton.
7. добавьте в панель управления кпопку «Текущие объекты». По нажатию на эту кнопку появляется модальное диалоговое окно, содержащее список всех «живых» объектов на момент нажатия со временем их рождения (время рождения – ключ). В класс диалогового окна должна передаваться коллекция с хранением объектов по времени рождения. Типы коллекций задаются вариантом.

***Вариант 1***

Коллекция для хранения объектов: Vector

Коллекция для хранения и поиска уникальных идентификаторов: HashSet

Коллекция для хранения времени рождения объектов: TreeMap

***Вариант 2***

Коллекция для хранения объектов: ArrayList

Коллекция для хранения и поиска уникальных идентификаторов: HashSet

Коллекция для хранения времени рождения объектов: TreeMap

***Вариант 3***

Коллекция для хранения объектов: LinkedList

Коллекция для хранения и поиска уникальных идентификаторов: HashSet

Коллекция для хранения времени рождения объектов: TreeMap

***Вариант 4***

Коллекция для хранения объектов: Vector

Коллекция для хранения и поиска уникальных идентификаторов: TreeSet

Коллекция для хранения времени рождения объектов: HashMap

***Вариант 5***

Коллекция для хранения объектов: ArrayList

Коллекция для хранения и поиска уникальных идентификаторов: TreeSet

Коллекция для хранения времени рождения объектов: HashMap

***Вариант 6***

Коллекция для хранения объектов: LinkedList

Коллекция для хранения и поиска уникальных идентификаторов: TreeSet

Коллекция для хранения времени рождения объектов: HashMap

***Вариант 7***

Коллекция для хранения объектов: Vector

Коллекция для хранения и поиска уникальных идентификаторов: HashSet

Коллекция для хранения времени рождения объектов: TreeMap

***Вариант 8***

Коллекция для хранения объектов: ArrayList

Коллекция для хранения и поиска уникальных идентификаторов: HashSet

Коллекция для хранения времени рождения объектов: TreeMap

***Вариант 9***

Коллекция для хранения объектов: LinkedList

Коллекция для хранения и поиска уникальных идентификаторов: HashSet

Коллекция для хранения времени рождения объектов: TreeMap

***Вариант 10***

Коллекция для хранения объектов: Vector

Коллекция для хранения и поиска уникальных идентификаторов: TreeSet

Коллекция для хранения времени рождения объектов: HashMap

***Вариант 11***

Коллекция для хранения объектов: ArrayList

Коллекция для хранения и поиска уникальных идентификаторов: TreeSet

Коллекция для хранения времени рождения объектов: HashMap

***Вариант 12***

Коллекция для хранения объектов: LinkedList

Коллекция для хранения и поиска уникальных идентификаторов: TreeSet

Коллекция для хранения времени рождения объектов: HashMap

***Вопросы для подготовки к защите***

1. Для чего используются классы-коллекции?
2. Что такое класс Vector? Для чего он используется? Каковы его достоинства и недостатки? Какая структура данных используется в классе Vector?
3. Назовите особенности организации класса Stack.
4. Для чего применяется класс Hashtable? Какая структура данных используется в классе Hashtable?
5. Что такое коэффициент загруженности?
6. Что такое емкость класса-коллекции?
7. Назначение и особенности применения класса Properties.
8. Расскажите об иерархии интерфейсов коллекций Java. Объясните назначение каждого интерфейса.
9. Каково назначение интерфейса Collection?
10. Опишите возможности применения интерфейсов Map, Set и List.
11. Для чего применяются интерфейсы Iterator и ListIterator?
12. Опишите классы ArrayList и LinkedList.
13. Опишите классы HashSet, TreeSet.
14. Опишите классы HashMap, TreeMap.
15. Какие структуры данных используются в этих классах?
16. Какие алгоритмы для обработки коллекций существуют в Java и как их использовать?

#### Лабораторная работа 4

**Темы**: ***Многопотоковые приложения.***

#### Практические задания

1. Изучить особенности реализации и работы потоков в Java, управлением приоритетами потоков и синхронизацией потоков.
2. Доработать программу, созданную в лабораторной работу № 3:
3. создать абстрактный класс BaseAI, описывающий «интеллектуальное поведение» объектов. Класс должен создавать поток, обеспечивающий движения объектов коллекции;
4. реализовать класс BaseAI для каждого из видов объекта, включив в него поведение, описанное в индивидуальном задании по варианту;
5. синхронизовать работу потоков расчета интеллекта объектов, их рисования и генерации новых объектов. Рисование должно остаться в основном потоке;
6. добавить в панель управления кнопки для остановки и возобновления работы интеллекта каждого вида объектов. Реализовать через засыпание/пробуждение потоков (методы wait() и notify());
7. добавить в панель управления выпадающие списки для выставления приоритетов каждого из потоков.

***Вариант 1***

1. Муравьи-рабочие двигаются в один из углов области их обитания (например, [0;0]) по прямой со скоростью V, а затем возвращаться обратно в точку своего рождения с той же скоростью.
2. Муравьи-воины двигаются по окружности с радиусом R со скоростью V.

***Вариант 2***

1. Пчелы-рабочие двигаются в один из углов области их обитания (например, [0;0]) по прямой со скоростью V, а затем возвращаться обратно в точку своего рождения с той же скоростью.
2. Трутни двигаются хаотично со скоростью V. Хаотичность достигается случайной сменой направления движения раз в N секунд.

***Вариант 3***

1. Золотые рыбки двигаются по оси X от одного края области обитания до другого со скоростью V.
2. Гуппи двигаются по оси Y от одного края области обитания до другого со скоростью V.

***Вариант 4***

1. Обыкновенные кролики двигаются хаотично со скоростью V. Хаотичность достигается случайной сменой направления движения раз в N секунд.
2. Альбиносы двигаются по оси X от одного края области обитания до другого со скоростью V.

***Вариант 5***

1. Грузовые машины двигаются в левую верхнюю четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке 0;0, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если машина сгенерировалась сразу в этой области, то она никуда не движется. По прибытии в конечную точку машина больше не движется.
2. Легковые машины двигаются в нижнюю правую четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке w/2;h/2, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если машина сгенерировалась сразу в этой области, то она никуда не движется. По прибытии в конечную точку машина больше не движется.

***Вариант 6***

1. Разработчики двигаются хаотично со скоростью V. Хаотичность достигается случайной сменой направления движения раз в N секунд.
2. Менеджеры двигаются по окружности с радиусом R со скоростью V.

***Вариант 7***

1. Капитальные дома двигаются (в городах будущего и не такое возможно) в левую верхнюю четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке 0;0, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если дом сгенерировался сразу в этой области, то он никуда не движется. По прибытии в конечную точку дом больше не движется.
2. Деревянные дома после генерации начинают двигаться в нижнюю правую четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке w/2;h/2, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если дом сгенерировался сразу в этой области, то он никуда не движется. По прибытии в конечную точку дом больше не движется.

***Вариант 8***

1. Автомобили двигаются по оси X от одного края области симуляции до другого со скоростью V.
2. Мотоциклы двигаются по оси Y от одного края области симуляции до другого со скоростью V.

***Вариант 9***

1. Студенты двигаются хаотично со скоростью V. Хаотичность достигается случайной сменой направления движения раз в N секунд.
2. Студентки двигаются по окружности с радиусом R со скоростью V.

***Вариант 10***

1. Юр. лица двигаются в левую верхнюю четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке 0;0, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если запись сгенерировалась сразу в этой области, то она никуда не движется. По прибытии в конечную точку запись больше не движется.
2. Физ. лица двигаются в нижнюю правую четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке w/2;h/2, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если запись сгенерировалась сразу в этой области, то она никуда не движется. По прибытии в конечную точку запись больше не движется.

***Вариант 11***

Кошки двигаются по периметру области симуляции со скоростью V1 . Собаки их догоняют со скоростью V2.

***Вариант 12***

Взрослые и птенцы двигаются в одном направлении (в стае). Направление движения меняется с периодом T и выбирается случайно.

***Вопросы для подготовки к защите***

1. Что такое процесс и поток (нить)?
2. Чем определяется порядок передачи управления потокам?
3. Какие есть способы реализации многозадачности в Java?
4. Что необходимо сделать для создания подкласса потоков (подкласса Thread)?
5. Когда запускается на выполнение метод run() подкласса Thread?
6. Какими методами класса Thread необходимо запускать поток на выполнение и останавливать его?
7. Что необходимо сделать для реализации классом интерфейса Runnable?
8. В каких состояниях может находиться поток?
9. Какой поток считается новым, выполняемым и завершенным?
10. В каких ситуациях поток является невыполняемым?
11. Что такое приоритет потока? На что он влияет?
12. Когда возникают исключительные ситуации при работе с потоками?
13. Что такое группы потоков и чем они полезны?
14. Что такое родовая группа потоков и главная группа потоков?
15. Что такое синхронизация потоков?
16. Что такое монитор? Как можно управлять монитором?
17. Что такое потоки-демоны? Чем они отличаются от рабочих потоков?
18. Как передаются данные между потоками в многопоточном приложении?
19. Что такое пулы потоков? Каков принцип их использования?

#### Лабораторная работа 5

**Темы**: ***Потоки ввода-вывода.***

***Сериализация объектов в файл***

#### Практические задания

1. Изучить особенности реализации системы ввода-вывода в Java.
2. Доработать программу, созданную в лабораторной работе № 4:
3. добавить в главное меню команду «Консоль». По этой команде должно появляться немодальное диалоговое окно с многострочным текстовым полем, занимающим всю область окна. В это окно можно вводить команды по варианту. В это же окно выводится реакция программы на команду;
4. для передачи команды в обрабатывающий поток использовать каналы ввода-вывода;
5. создать конфигурационный файл для программы. В конфигурационный файл должны сохраняться все настройки симуляции, т.е. все данные и состояния, которые задаются в панели управления программы. Конфигурационный файл должен читаться при запуске программы и записываться при выходе. Формат файла текстовый;
6. добавить в главное меню пункты «Загрузить» и «Сохранить». Команда «Сохранить» вызывает сериализацию всех «живых» объектов в ней. Команда «Загрузить» останавливает текущую симуляцию (если симуляция запущена) и загружает объекты из выбранного файла. Не забудьте скорректировать время рождения объектов. После открытия симуляцию можно запустить, загруженные объекты должны вести себя естественно;
7. использовать стандартные файловые диалоги.

***Вариант 1***

Реализовать в консоли команды «Старт» и «Стоп» симуляции. Кнопки в интерфейсе должны вести себя так же, как если бы нажимали их, а не исполняли команды (блокироваться по очереди).

***Вариант 2***

Реализовать в консоли команду «Вернуть количество живых пчел-рабочих/трутней». Как параметр в команду должен передаваться идентификатор вида объекта.

***Вариант 3***

Реализовать в консоли команды «Установить вероятность рождения золотых рыбок» и «Получить вероятность рождения золотых рыбок». Как параметр в команду установки должно передаваться значение вероятности. Полученная вероятность должна выводиться на консоль.

***Вариант 4***

Реализовать в консоли команду «Сократить число кроликов-альбиносов на N%». Как параметр в команду должно передаваться значение N%.

***Вариант 5***

Реализовать в консоли команды «Остановить интеллектуальное поведение объектов» и «Продолжить интеллектуальное поведение объектов». Команда «Остановить» - останавливает поток расчета интеллекта объектов, а команда «Продолжить» - возобновляет расчет после остановки.

***Вариант 6***

Реализовать в консоли команду «Уволить всех менеджеров» и «Нанять N новых менеджеров». Первая команда удаляет всех менеджеров из симуляции (новые продолжают генерироваться), вторая – генерирует N новых менеджеров. N – параметр команды.

***Вариант 7***

Реализовать в консоли команды «Установить вероятность генерации капитальных домов» и «Получить вероятность генерации капитальных домов». Как параметр в команду установки должно передаваться значение вероятности. Полученная вероятность должна выводиться на консоль.

***Вариант 8***

Реализовать в консоли команду «Сократить число мотоциклов на N%». Как параметр в команду должно передаваться значение N%.

***Вариант 9***

Реализовать в консоли команды «Показывать время симуляции» и «Скрывать время симуляции». Чекбокс в интерфейсе должен вести себя так же, как если бы использовали его, а не исполняли команды (галочка появляется/исчезает).

***Вариант 10***

Реализовать в консоли команду «Вернуть количество существующих юридических/физических лиц в картотеке». Как параметр в команду должен передаваться идентификатор вида объекта.

***Вариант 11***

Реализовать в консоли команды «Остановить генерацию ХХХ объектов» и «Продолжить генерацию ХХХ объектов». Чекбокс в интерфейсе должен вести себя так же, как если бы использовали его, а не исполняли команды (галочка появляется/исчезает).

***Вариант 12***

Реализовать в консоли команду «Изменить процент птенцов».

***Вопросы для подготовки к защите***

1. Что такое потоки ввода-вывода и для чего они нужны?
2. Какие классы Java являются базовыми для работы с потоками?
3. В чем разница между байтовыми и символьными потоками?
4. Какие стандартные потоки ввода-вывода существуют в Java, каково их назначение? На базе каких классов создаются стандартные потоки?
5. Чем является поток System.in, System.out, System.err? Какими методами чаще всего пользуются при работе с этим потоком?
6. Как создать файловый поток для чтения и записи данных?
7. В чем заключается особенность создания потока, связанного с локальным файлом?
8. Как создать поток для форматированного обмена данными, связанного с локальным файлом?
9. Как добавить буферизацию для потока форматированного обмена данными, связанного с локальным файлом?
10. За счет чего буферизация ускоряет работу приложений с потоками?
11. Когда применяется принудительный сброс буферов?
12. Для выполнения каких операций применяется класс File?
13. Для чего предназначен класс RandomAccessFile? Чем он отличается от потоков ввода и вывода?
14. Что такое сериализация объектов? Что такое десериализация объектов?
15. Как объявить класс сериализуемым?
16. Какие поля класса не сериализуются?
17. Как можно реализовать свой алгоритм сериализации.

#### Лабораторная работа 6

**Темы**: ***Сетевые «клент-серверные» приложения***

**Практические задания**

1. Изучить особенности реализации сетевых приложений в Java.
2. Доработать программу, созданную в лабораторной работе № 5:
3. создать отдельное консольное приложение, которое будет играть роль TCP-сервера. Клиентом будет приложение, которое было создано в предыдущих работах;
4. TCP-сервер должен ожидать подключения клиентов и выдавать вновь подключенному клиенту список уже подключенных. Необходимо также внести изменения в интерфейс клиента, так чтобы в панели управления отображался список всех подключенных к серверу клиентов. При отсоединении клиентов или при подключении новых список должен обновляться;
5. запрограммировать специальное взаимодействие по TCP с другими клиентами через сервер по варианту;

***Вариант 1***

Реализовать выбор из одного подключенного клиента N случайных объектов и поместить в текущую симуляцию.

***Вариант 2***

Реализовать возможность отправить N случайных объектов из текущей симуляции другому подключенному клиенту.

***Вариант 3***

Реализовать возможность получения и установки настроек симуляции таких же как у одного из подключенных клиентов.

***Вариант 4***

Реализовать возможность передачи своих настроек симуляции одному из подключенных клиентов.

***Вариант 5***

Реализовать возможность обмена всех объектов одного вида на объекты такого же вида из другого подключенного клиента (например, все легковые машины из одной симуляции переходят в другую, а все легковые машины, что были в той другой симуляции, переходят в первую).

***Вариант 6***

Реализовать возможность обмена всех объектов одного вида на объекты другого вида из другого подключенного клиента (например, все менеджеры из одной симуляции переходят в другую, а все разработчики, что были в той другой симуляции, переходят в первую).

***Вариант 7***

Реализовать возможность скопировать и добавить в симуляцию все объекты одного из подключенных клиентов.

***Вариант 8***

Реализовать возможность обменяться всеми объектами с одним из подключенных клиентов.

***Вариант 9***

Реализовать возможность синхронной остановки и запуска симуляции на текущем и одном из подключенных клиентов, т.е. при остановке или запуске симуляции выбранный клиент должен также остановить или запустить симуляцию.

***Вариант 10***

Реализовать возможность синхронной установки вероятностей появления физических и юридических лиц на текущем и одном из подключенных клиентов, т.е. при установке этих параметров на выбранном клиенте должны выставиться такие же параметры.

***Вариант 11***

Реализовать возможность обмена объектами с выбранным клиентом. У одного собираются все кошки, у другого все собаки.

***Вариант 12***

Реализовать передачу от клиента другому выбранному клиенту числа, получив которое число птиц его коллекции уменьшается на заданный процент.

***Вопросы для подготовки к защите***

1. Что такое сокеты?
2. Какие типы сокетов существуют, чем они отличаются друг от друга?
3. Какое преимущество имеют потоковые сокеты?
4. Что такое IP-адрес и доменный адрес узла (хоста)?
5. Какой класс Java используется для представления адреса хоста?
6. Какой класс Java предназначен для работы с IP-адресами?
7. Каким образом задается адрес узла при создании объекта, отвечающего за адрес IP?
8. Что такое localhost?
9. Что такое ТСР/IP ?
10. Как создается сокетное соединение «сервер-клиент»?
11. Какова последовательность действий приложений Java, необходимая для создания канала и передачи данных между клиентским и серверным приложением?
12. Можно ли оборачивать потоки ввода-вывода сокетов другими потоками ввода-вывода из java.io?
13. Почему сетевые программы в большинстве случаев являются многопоточными?
14. Каковы недостатки и преимущества дейтаграммных сокетов?
15. Что должны сделать приложения для работы с дейтаграммами?
16. Какие методы применяются для посылки и получения дейтаграмм?

#### Лабораторная работа 7

**Темы**: ***Организация в Java хранения данных в базе данных***

**Практические задания**

1. Изучить особенности реализации хранения данных в базе данных в приложениях Java.
2. Доработать программу, созданную в лабораторной работе № 5:

* Добавить в меню пункты сохранение и загрузку объектов в базу данных. Можно использовать любую СУБД.
* Загрузку и сохранение объектов только заданного типа.

***Вопросы для подготовки к защите***

1. Что такое JDBC API и когда его используют?
2. Что такое JDBC Driver и какие различные типы драйверов JDBC вы знаете?
3. Как JDBC API помогает достичь слабой связи между Java программой и JDBC Drivers API?
4. Что такое JDBC Connection? Покажите шаги для подключения программы к базе данных.
5. Как используется JDBC DriverManager class?
6. Как получить информацию о сервере базы данных из java программы?
7. Что такое JDBC Statement?
8. Какие различия между execute, executeQuery, executeUpdate?
9. Что такое JDBC PreparedStatement?
10. Что такое SQL?

#### Что такое JDBC ResultSet?

#### Какие существуют различные типы JDBC ResultSet?

1. Как обработать данные, получаемые в результате выполнения запроса?

#### Лабораторная работа 8

**Темы**: ***Generic-классы в java***

**Практические задания**

1. Изучить принципы построения классов и методов generic в Java.
2. Разработать программу, использующую классы объектов созданных в лабораторной работе № 1:
   1. разработать Generic-класс, позволяющий работать с объектами по варианту (добавлять, удалять и т.д.). В качестве параметра класс должен принимать различные объекты (использовать ограничения на тип);
   2. разработать метод Generic-класса позволяющий добавить в структуру коллекцию объектов. Протестируйте метод на коллекциях разных видов и посмотрите за ошибками компиляции, которые возникают при неправильном использовании generics;
   3. разработать статический generic-метод, который работает с разработанным Generic-классом. Действие, которое должен выполнять метод, задается индивидуальным вариантом. Для всех вариантов для ввода-вывода использовать консоль, созданную в лабораторной работе № 5.

***Вариант 1***

Структура данных generic-класса стек на списке.

Generic-метод возвращает копию входной группы с инвертированным порядком элементов.

***Вариант 2***

Структура данных generic-класса циклическая очередь.

Generic-метод принимает на вход 2 группы, объединяет группы и возвращает объединенную группу.

***Вариант 3***

Структура данных generic-класса односвязный список.

Generic-метод возвращает группу, в которой содержатся четные элементы из входной группы.

***Вариант 4***

Структура данных generic-класса двусвязный циклический список.

Generic-метод принимает на вход группу, создает ее копию, перемешивает в этой копии элементы случайным образом и возвращает ее.

***Вариант 5***

Структура данных generic-класса очередь.

Generic-метод удаляет из входной группы все элементы.

***Вариант 6***

Структура данных generic-класса односвязный циклический список.

Generic-метод возвращает группу, в которой содержатся нечетные элементы из входной группы.

***Вариант 7***

Структура данных generic-класса двусвязный список.

Generic-метод удаляет из входной группы все четные элементы.

***Вариант 8***

Структура данных generic-класса односвязный список.

Generic-метод принимает на вход 2 группы, и возвращает группу, которая содержит элементы, присутствующие и в первой, и во второй группе, т.е. возвращает пересечение двух групп.

***Вариант 9***

Структура данных generic-класса очередь на динамическом массиве.

Generic-метод удаляет из входной группы все нечетные элементы.

***Вариант 10***

Структура данных generic-класса двусвязный список.

Generic-метод принимает на вход 2 группы, и возвращает группу, которая содержит уникальные неповторяющиеся элементы из обеих групп (т.е. элементы, содержащиеся в первой группе, но не содержащиеся во второй, или содержащиеся во второй, но не содержащиеся в первой).

***Вариант 11***

Структура данных generic-класса двунаправленная очередь на динамических массивах.

Generic-метод принимает на вход 2 группы, и возвращает группу, которая содержит элементы двух групп.

***Вариант 12***

Структура данных generic-класса двунаправленная очередь на списках.

Generic-метод удаляет из входной группы элементы заданного типа.

***Вопросы для подготовки к защите***

1. Назначение generic.
2. В чем сходство и отличие шаблонов в С++ и generic в Java?
3. Что такое неопределенный тип? Приведите примеры его задания и использования.
4. Что такое ограничения на неопределенный тип? Когда их можно использовать приведите примеры.
5. Как можно задать ограничения на неопределенный тип?
6. Что такое методы generic? Зачем они применяются? Приведите примеры.
7. Каковы сложности при создании нового API с generic?
8. Как переделывать существующее API под generic?
9. Что такое стирание информации о типе при компиляции, и почему это применяется?