Test Data Builders

```
public class EntityBuilder {
    private Long entityId = 0L;
    private EntityType type = EntityType.CLOSED;
    private RelatedEntity relatedEntity;
    private List<SubEntity> subEntities = new ArrayList<SubEntity>();
    private Date createdDate = new Date();

    public static EntityBuilder anEntity() {
        return new EntityBuilder();
    }

    public Entity build() {
        Entity entity = new Entity();
        entity.setEntityId(entityId);
        entity.setType(type);
        entity.setRelatedEntity(relatedEntity);
        entity.setSubEntities(subEntities);
        return entity;
    }

    public EntityBuilder withEntityId(Long entityId) {
        this.entityId = entityId; return this;
    }

    public EntityBuilder with(EntityType type) {
        this.type = type; return this;
    }

    public EntityBuilder with(RelatedEntity relatedEntity) {
        this.relatedEntity = relatedEntity; return this;
    }

    public EntityBuilder with(RelatedEntityBuilder builder) {
        this.relatedEntity = builder.build(); return this;
    }

    public EntityBuilder with(SubEntity subEntity) {
        this.subEntities.add(subEntity); return this;
    }

    public EntityBuilder with(SubEntityBuilder builder) {
        this.subEntities.add(builder.build()); return this;
    }
    public EntityBuilder withCreatedDate(int day, int month, int year) {
        this.createdDate = TestUtils.createDate(day, month, year);
        return this;
    }
```

Fields are the same as entity's fields, just with default values
Relationships with other entities do not get initialized with default values, one-to-many relationships are initialized with empty collection

Static factory method to be used in tests in combination with static imports.
E.g.: anEntity.with…().build())
Instead of: new EntityBuilder().with…().build()

Here we construct the result object by populating all the properties of the entity to the default values or values set by calling with…() methods.

Setter for the property of the entity. Always returns the same instance of the builder to allow method chaining.

If property type is unique for this entity we can omit property name in method name. Method with() will be overloaded with every unique type

Setters for related entities are overloaded to take the builder of related entity. This allows to avoid multiple calls to build () method of each builder. E.g.:

```
anEntity().with(aRelatedEntity()
                .withName("related")
                .with(aSubRelatedEntity()))
          .build()
Instead of:
anEntity().with(aRelatedEntity()
                .withName("related")
                .with(aSubRelatedEntity()
                      .build())
                .build())
          .build()
```

Overload other with…() methods to use convenience utilities.

Inspired by "Growing Object-Oriented Software, Guided by Test" book by Nat Pryce and Steve Freeman
ISBN: 978-0321503626

Created by Stanislav Chekalin